NAVAL POSTGRADUATE SCHOOL

Monterey, California

1

THESIS

PART II

A DESIGN OF COMPUTER AIDED INSTRUCTIONS (CAI) FOR UNDIRECTED GRAPHS IN THE DISCRETE MATH TUTORIAL (DMT)

by

Atilla Bakan & Yavuz Bas

June 1990

Thesis Advisors: Hefner & Shing

91

```
outtextxy(2*x,16*y," A (currently largest labeled) in L");
outtextxy(2*x,17*y," and label them with k + 1 = 1");
Pause(7*x,24*y);
outtextxy(52*x,4*y,"B");
outtextxy(52*x,5*y,"E");
/*****************************************************************/
outtextxy(58*x,4*y,"B <- 1");
outtextxy(58*x,5*y,"E <- 1");
outtextxy(22*x,4*y,"(1)");
outtextxy(8*x,15*y/2,"(1)");
/*****************************************************************/
outtextxy(2*x,19*y,". Put the edges connecting these ");
outtextxy(2*x,20*y," vertices to A in the tree T.");
Pause(7*x,24*y);
outtextxy(72*x,4*y,"(A,B)");
outtextxy(72*x,5*y,"(A,E)");
setcolor(backcolor);
moveto(20*x,4*y);  lineto(10*x,4*y); lineto(10*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,4*y); lineto(10*x,4*y);  /* add (A,B) to T */
lineto(10*x,7*y);                    /* add (A,D) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,22*y,". Increment k and go to Step 2.");
Pause(7*x,24*y);
outtextxy(45*x,4*y,"1");
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Since L does not contain all the vertices");
outtextxy(2*x,16*y," of the graph G, find all the unlabeled");
outtextxy(2*x,17*y," vertices adjacent to those currently");
outtextxy(2*x,18*y," having largest labeles and label them");
outtextxy(2*x,19*y," with k + 1 = 2");
```

```
Pause(7*x,24*y);
outtextxy(52*x,6*y,"F");
/*********************************************************************/
outtextxy(58*x,6*y,"F <- 2");
outtextxy(21*x,27*y/4,"(2)");
/*********************************************************************/
outtextxy(2*x,20*y,". As you see there are more than one edge");
outtextxy(2*x,21*y," conneting F to those labeled with 1, so ");
outtextxy(2*x,22*y," choose one of them arbitrarily.");
Pause(7*x,24*y);
outtextxy(72*x,6*y,"(B,F)");
setcolor(backcolor);
moveto(20*x,4*y); lineto(20*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,4*y); lineto(20*x,7*y); /* add (B,F) to T */
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(2*x,23*y,". Increment k and go to Step 2.");
Pause(7*x,24*y);
outtextxy(45*x,6*y,"2");
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(2*x,15*y,". Since L does not contain all the vertices");
outtextxy(2*x,16*y," of The graph G, find all the unlabeled");
outtextxy(2*x,17*y," vertices adjacent to those currently");
outtextxy(2*x,18*y," having largest labeles and label them");
outtextxy(2*x,19*y," with k + 1 = 3");
Pause(7*x,24*y);
outtextxy(52*x,7*y,"G");
outtextxy(52*x,8*y,"I");
/*********************************************************************/
outtextxy(58*x,7*y,"G <- 3");
outtextxy(58*x,8*y,"I <- 3");
outtextxy(16*x,10*y,"(3)");
outtextxy(26*x,27*y/4,"(3)");
/*****************************************************************/
outtextxy(2*x,20*y,". Put the edges connecting these ");
outtextxy(2*x,21*y," vertices to the vertices labeled");
outtextxy(2*x,22*y," with 2, in the tree T.");
Pause(7*x,24*y);
outtextxy(72*x,7*y,"(F,G)");
outtextxy(72*x,8*y,"(F,I)");
setcolor(backcolor);
moveto(20*x,10*y);  lineto(20*x,7*y); lineto(30*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,10*y); lineto(20*x,7*y);  /* add (F,I) to T */
lineto(30*x,7*y);                     /* add (F,G) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,23*y,". Increment k and go to Step 2.");
Pause(7*x,24*y);
outtextxy(45*x,7*y,"3");
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Since L does not contain all the vertices");
outtextxy(2*x,16*y," of The graph G, find all the unlabeled");
outtextxy(2*x,17*y," vertices adjacent to those currently");
outtextxy(2*x,18*y," having largest labeles and label them");
outtextxy(2*x,19*y," with k + 1 = 4");
Pause(7*x,24*y);
outtextxy(52*x,9*y,"C");
outtextxy(52*x,10*y,"H");
outtextxy(52*x,11*y,"J");
/*****************************************************************/
outtextxy(58*x,9*y,"C <- 4");
outtextxy(58*x,10*y,"H <- 4");
outtextxy(58*x,11*y,"J <- 4");
outtextxy(38*x,15*y/2,"(4)");
outtextxy(26*x,4*y,"(4)");
outtextxy(32*x,10*y,"(4)");
/*****************************************************************/
outtextxy(2*x,20*y,". Put the edges connecting these ");
outtextxy(2*x,21*y,"  vertices to the vertices labeled");
outtextxy(2*x,22*y,"  with 3, in the tree T.");
Pause(7*x,24*y);
outtextxy(72*x,9*y,"(G,J) (Why not");
outtextxy(72*x,19*y/2,"       (I,J) ?)");
outtextxy(72*x,10*y,"(G,C)");
outtextxy(72*x,11*y,"(G,H)");
setcolor(backcolor);
moveto(40*x,7*y);  lineto(30*x,7*y); lineto(30*x,4*y);
moveto(30*x,7*y);  lineto(30*x,10*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(40*x,7*y); lineto(30*x,7*y);  /* add (G,H) to T */
lineto(30*x,4*y);                    /* add (G,C) to T */
moveto(30*x,7*y); lineto(30*x,10*y); /* add (G,J) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,23*y,". Increment k and go to Step 2.");
Pause(7*x,24*y);
outtextxy(45*x,9*y,"4");
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Since L does not contain all the vertices");
outtextxy(2*x,16*y,"  of The graph G, find all the unlabeled");
outtextxy(2*x,17*y,"  vertices adjacent to those currently");
outtextxy(2*x,18*y,"  having largest labeles and label them");
outtextxy(2*x,19*y,"  with k + 1 = 5");
Pause(7*x,24*y);
```

```
    outtextxy(52*x,12*y,"D");
    /**********************************************************************/
    outtextxy(58*x,12*y,"D <- 5");
    outtextxy(41*x,4*y,"(5)");
    /**********************************************************************/
    outtextxy(2*x,20*y,". Put the edges connecting this ");
    outtextxy(2*x,21*y,"  vertice to one of those labeled");
    outtextxy(2*x,22*y,"  with 4, in the tree T.");
    Pause(7*x,24*y);
    outtextxy(72*x,12*y,"(H,D)");
    setcolor(backcolor);
    moveto(40*x,7*y);  lineto(40*x,4*y);
    setcolor(forecolor);
    setlinestyle(3,0,3);
    moveto(40*x,7*y); lineto(40*x,4*y);  /* add (H,D) to T */
    setlinestyle(0,0,3);
    /**********************************************************************/
    outtextxy(2*x,23*y,". Increment k and go to Step 2.");
    Pause(7*x,24*y);
    outtextxy(45*x,12*y,"5");
    setcolor(backcolor);
    bar(3*x/2,59*y/4,55*x,49*y/2);
    setcolor(forecolor);
    /**********************************************************************/
    outtextxy(2*x,15*y,". As you see L contains all the vertices");
    outtextxy(2*x,16*y,"  of the graph G. This means we are done.");
    /**********************************************************************/
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM    : exspan6.c
  AUTHOR      :  Atilla BAKAN
  DATE        : Apr. 18, 1990
  REVISED     : Apr. 18, 1990

  DESCRIPTION : This routine draws the example graph for a graph which cannot
                have spanning tree.


  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                      /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk           find_t
  #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph   (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions     */
static void exer             (void);

/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;



/*********************************************************************/
/* This function is used for including drivers to the executable code         */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*********************************************************************/
/* This fuction initializes the necessary graphical routines        */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*******************************************************************/
  settext();
  /*******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      }
  forecolor = WHITE;
 }
```

```
/***********************************************************************/
/* This function sets the text default values                          */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}


/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>> PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
  }


/***********************************************************************/
/* main routine that calls exer routine                                */
/***********************************************************************/
void main()
{
  exer();
}
```

```
/**********************************************************************/
/* This routine illustrates a graph which cannot have spanning tree.          */
/**********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE SPAN_6");
  /********************************************************************/
  outtextxy(2*x,2*y,"The graph in this figure does not have a spanning tree because
                         it is not");
  outtextxy(2*x,3*y,"possible to choose edges that connect all the vertices of G. In
                         particular,");
  outtextxy(2*x,4*y,"we cannot find edges of G that can be used to make a path from
                         A to D.");
  /********************************************************************/
  pieslice(35*x,13*y,0,359,2);
  pieslice(55*x,13*y,0,359,2);
  pieslice(45*x,9*y,0,359,2);
  pieslice(45*x,17*y,0,359,2);
  moveto(35*x,13*y);  lineto(55*x,13*y);
  lineto(45*x,9*y);  lineto(35*x,13*y);
  outtextxy(45*x,17*y/2,"A");
  outtextxy(33*x,13*y,"B");
  outtextxy(56*x,13*y,"C");
  outtextxy(46*x,17*y,"D");
  /********************************************************************/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
}
```

```
/* PROGRAM   : qs421.c
   AUTHOR    : Atilla BAKAN
   DATE      : Mar. 22, 1990
   REVISED   : Apr. 17, 1990

   DESCRIPTION : This program contains the first exercise about the
                 spanning trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                  /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions           */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);
static void example          (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/*********************************************************************
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;


/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*****************************************************************/
/* This function is used for including drivers to the executable code     */
/*****************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*****************************************************************/
/* This fuction initializes the necessary graphical routines              */
/*****************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***************************************************************/
  settext();
  /***************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/*****************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)          {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
      case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
      case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}



/*********************************************************************/
/* This function sets the text default values                        */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen               */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/*************************************************************************/
/* main routine that calls exer routine                                 */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/**********************************************************************/
/* Routine that asks the question, then depending on the user's answer        */
/* makes necessary explanations                                                      */
/**********************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  1");
  outtextxy(2*x,2*y,"Use the breadth first search algorithm to find a minimal
                                spanning tree.");
  outtextxy(2*x,3*y,"(Start at A. If there is a choice of edges select edges according
                                to");
  outtextxy(2*x,4*y,"alphabetical order.)");
  pieslice(20*x,5*y,0,359,2);       /* A */
  pieslice(30*x,5*y,0,359,2);       /* B */
  pieslice(20*x,13*y/2,0,359,2);    /* C */
  pieslice(30*x,13*y/2,0,359,2);    /* D */
  pieslice(40*x,13*y/2,0,359,2);    /* E */
  pieslice(30*x,8*y,0,359,2);       /* F */
  pieslice(40*x,8*y,0,359,2);       /* G */
  pieslice(50*x,8*y,0,359,2);       /* H */
  pieslice(40*x,19*y/2,0,359,2);    /* I */
  pieslice(50*x,19*y/2,0,359,2);    /* J */
  pieslice(60*x,19*y/2,0,359,2);    /* K */
  pieslice(50*x,11*y,0,359,2);      /* L */
  pieslice(60*x,11*y,0,359,2);      /* M */
  outtextxy(20*x,9*y/2,"A");
  outtextxy(30*x,9*y/2,"B");
  outtextxy(18*x,13*y/2,"C");
  outtextxy(31*x,25*y/4,"D");
  outtextxy(41*x,13*y/2,"E");
outtextxy(28*x,8*y,"F");
outtextxy(41*x,31*y/4,"G");
outtextxy(51*x,8*y,"H");
outtextxy(38*x,19*y/2,"I");
outtextxy(51*x,37*y/4,"J");
outtextxy(60*x,9*y,"K");
outtextxy(48*x,11*y,"L");
outtextxy(60*x,23*y/2,"M");
moveto(20*x,5*y);  lineto(30*x,5*y);  lineto(30*x,8*y);
lineto(50*x,8*y);  lineto(50*x,11*y);  lineto(60*x,11*y);
moveto(20*x,5*y);  lineto(20*x,13*y/2); lineto(40*x,13*y/2);
lineto(40*x,19*y/2); lineto(60*x,19*y/2); lineto(60*x,11*y);
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
    switch (Ch)        {
    case 'a': outtextxy(47*x,19*y,"a");
      outtextxy(52*x,19*y,"You want to see the algorithm ");
      outtextxy(52*x,20*y,"again. Press any key to continue.");
      Pause(30*x,24*y);
```

```
       setcolor(backcolor);
       bar(50*x,37*y/2,179*x/2,21*y);
       bar(2*x,13*y,179*x/2,49*y/2);
       setcolor(forecolor);
       show_alg();
       break;
     case 'b': outtextxy(47*x,19*y,"b");
       outtextxy(52*x,19*y,"You want to compare your solu-");
       outtextxy(52*x,20*y,"tion with ours. So press any  ");
       outtextxy(52*x,21*y,"key to see it.");
       Pause(30*x,24*y);
       setcolor(backcolor);
       bar(50*x,37*y/2,179*x/2,22*y);
       bar(2*x,13*y,179*x/2,49*y/2);
       setcolor(forecolor);
       compare_solutions();
       break;
     case 'c': outtextxy(47*x,19*y,"c");
       outtextxy(52*x,19*y,"You want to see step by step");
       outtextxy(52*x,20*y,"solution. So press any key to ");
       outtextxy(52*x,21*y,"continue.");
       Pause(30*x,24*y);
       setcolor(backcolor);
       bar(50*x,37*y/2,179*x/2,22*y);
       bar(2*x,13*y,179*x/2,49*y/2);
       setcolor(forecolor);
       step_solution();
       break;
     case 'd': outtextxy(47*x,19*y,"d");
       confirm_exit();
       break;
     default  : break;
    }
  }
  closegraph();
}
```

```
/***********************************************************************/
/* This routine gives the step by step solution to the exercise        */
/***********************************************************************/
static void step_solution(void)
{

  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  outtextxy(64*x,5*y,"k");
  outtextxy(70*x,5*y,"L");
  outtextxy(75*x,5*y,"Label");
  outtextxy(86*x,5*y,"T");
  moveto(62*x,11*y/2);  lineto(67*x,11*y/2);
  moveto(68*x,11*y/2);  lineto(73*x,11*y/2);
  moveto(74*x,11*y/2);  lineto(165*x/2,11*y/2);
  moveto(84*x,11*y/2);  lineto(89*x,11*y/2);
  /*********************************************************************/
  outtextxy(70*x,6*y,"A");
  outtextxy(64*x,6*y,"0");
  outtextxy(75*x,6*y,"A <- 0");
  outtextxy(16*x,5*y,"(0)");
  /*********************************************************************/
  Pause(30*x,24*y);
  outtextxy(70*x,7*y,"B");
  outtextxy(70*x,8*y,"C");
  outtextxy(75*x,7*y,"B <- 1");
  outtextxy(84*x,7*y,"(A,B)");
  outtextxy(31*x,5*y,"(1)");
  outtextxy(75*x,8*y,"C <- 1");
  outtextxy(84*x,8*y,"(A,C)");
  outtextxy(19*x,7*y,"(1)");
  Pause(30*x,24*y);
  setcolor(backcolor);
  moveto(20*x,5*y);  lineto(30*x,5*y);
moveto(20*x,5*y); lineto(20*x,13*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,5*y); lineto(30*x,5*y);    /* add (A, B) to T */
moveto(20*x,5*y); lineto(20*x,13*y/2); /* add (A, C) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,7*y,"1");
/*********************************************************************/
outtextxy(70*x,9*y,"D");
outtextxy(75*x,9*y,"D <- 2");
outtextxy(84*x,9*y,"(B,D)");
outtextxy(67*x/2,25*y/4,"(2)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(30*x,5*y); lineto(30*x,13*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,5*y); lineto(30*x,13*y/2); /* add (B, D) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,9*y,"2");
/*********************************************************************/
outtextxy(70*x,10*y,"E");
outtextxy(70*x,11*y,"F");
outtextxy(75*x,10*y,"E <- 3");
outtextxy(84*x,10*y,"(D,E)");
outtextxy(43*x,13*y/2,"(3)");
outtextxy(75*x,11*y,"F <- 3");
outtextxy(84*x,11*y,"(D,F)");
outtextxy(29*x,17*y/2,"(3)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(30*x,13*y/2); lineto(30*x,8*y);
moveto(30*x,13*y/2); lineto(40*x,13*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,13*y/2); lineto(30*x,8*y);     /* add (D, F) to T */
moveto(30*x,13*y/2); lineto(40*x,13*y/2); /* add (D, E) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,10*y,"3");
/*****************************************************************/
outtextxy(70*x,12*y,"G");
outtextxy(75*x,12*y,"G <- 4");
outtextxy(84*x,12*y,"(E,G)");
outtextxy(43*x,31*y/4,"(4)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(40*x,13*y/2); lineto(40*x,8*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,13*y/2); lineto(40*x,8*y);     /* add (E, G) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,12*y,"4");
/*****************************************************************/
outtextxy(70*x,13*y,"H");
outtextxy(70*x,14*y,"I");
outtextxy(75*x,13*y,"H <- 5");
outtextxy(84*x,13*y,"(G,H)");
outtextxy(52*x,8*y,"(5)");
outtextxy(75*x,14*y,"I <- 5");
outtextxy(84*x,14*y,"(G,I)");
outtextxy(39*x,10*y,"(5)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(40*x,8*y);  lineto(50*x,8*y);
moveto(40*x,8*y);  lineto(40*x,19*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,8*y);  lineto(50*x,8*y);     /* add (G, H) to T */
moveto(40*x,8*y);  lineto(40*x,19*y/2);  /* add (G, I) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,13*y,"5");
/*****************************************************************/
outtextxy(70*x,15*y,"J");
outtextxy(75*x,15*y,"J <- 6");
outtextxy(84*x,15*y,"(H,J)");
outtextxy(46*x,9*y,"(6)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(50*x,8*y);  lineto(50*x,19*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,8*y);  lineto(50*x,19*y/2);     /* add (H, J) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,15*y,"6");
```

```
/*****************************************************************/
outtextxy(70*x,16*y,"K");
outtextxy(70*x,17*y,"L");
outtextxy(75*x,16*y,"K <- 7");
outtextxy(84*x,16*y,"(J,K)");
outtextxy(56*x,9*y,"(7)");
outtextxy(75*x,17*y,"L <- 7");
outtextxy(84*x,17*y,"(J,L)");
outtextxy(49*x,23*y/2,"(7)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(50*x,19*y/2); lineto(60*x,19*y/2);
moveto(50*x,19*y/2); lineto(50*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,19*y/2); lineto(60*x,19*y/2);    /* add (J, K) to T */
moveto(50*x,19*y/2); lineto(50*x,11*y);      /* add (J, L) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,16*y,"7");
/*****************************************************************/
outtextxy(70*x,18*y,"M");
outtextxy(75*x,18*y,"M <- 8");
outtextxy(84*x,18*y,"(K,M)");
outtextxy(56*x,23*y/2,"(8)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(60*x,19*y/2); lineto(60*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(60*x,19*y/2); lineto(60*x,11*y);      /* add (K, M) to T */
setlinestyle(0,0,3);
outtextxy(64*x,18*y,"8");
```

```
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(65*x,20*y,"We are done.");
/***********************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);          /* Clean the game field again */
bar(3*x/2,17*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/***********************************************************************/
pieslice(20*x,5*y,0,359,2);       /* A */  /* redraw the graph */
pieslice(30*x,5*y,0,359,2);       /* B */
pieslice(20*x,13*y/2,0,359,2);  /* C */
pieslice(30*x,13*y/2,0,359,2);  /* D */
pieslice(40*x,13*y/2,0,359,2);  /* E */
pieslice(30*x,8*y,0,359,2);       /* F */
pieslice(40*x,8*y,0,359,2);       /* G */
pieslice(50*x,8*y,0,359,2);       /* H */
pieslice(40*x,19*y/2,0,359,2);  /* I */
pieslice(50*x,19*y/2,0,359,2);  /* J */
pieslice(60*x,19*y/2,0,359,2);  /* K */
pieslice(50*x,11*y,0,359,2);      /* L */
pieslice(60*x,11*y,0,359,2);      /* M */
/***********************************************************************/
outtextxy(20*x,9*y/2,"A");
outtextxy(30*x,9*y/2,"B");
outtextxy(18*x,13*y/2,"C");
outtextxy(31*x,25*y/4,"D");
outtextxy(41*x,13*y/2,"E");
outtextxy(28*x,8*y,"F");
outtextxy(41*x,31*y/4,"G");
outtextxy(51*x,8*y,"H");
outtextxy(38*x,19*y/2,"I");
outtextxy(51*x,37*y/4,"J");
outtextxy(60*x,9*y,"K");
```

```
  outtextxy(48*x,11*y,"L");
  outtextxy(60*x,23*y/2,"M");
 /*******************************************************************/
  moveto(20*x,5*y);   lineto(30*x,5*y);   lineto(30*x,8*y);
  lineto(50*x,8*y);   lineto(50*x,11*y);  lineto(60*x,11*y);
  moveto(20*x,5*y);   lineto(20*x,13*y/2); lineto(40*x,13*y/2);
  lineto(40*x,19*y/2); lineto(60*x,19*y/2); lineto(60*x,11*y);
}

/*********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
   switch (ch)         {
    case 'y': in_the_exercise = 0;
            break;
    case 'Y': in_the_exercise = 0;
            break;
    case 'n': setcolor(backcolor);
            bar(46*x,37*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
```

```
            bar(46*x,37*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;
        default : break;
        }
}
```

```
/* PROGRAM   : qs422.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 7, 1990
   REVISED   : Apr. 7, 1990

   DESCRIPTION : This program contains the second exercise about the
                 spanning trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                    /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk           find_t
  #define ff_name         name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);
/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;



/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code      */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines                */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*******************************************************************/
  settext();
  /*******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/**********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)         {
     case 'y': closegraph();
```

```
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/************************************************************************/
/* This function sets the text default values                          */
/************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen               */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/**********************************************************************/
/* main routine that calls exer routine                                 */
/**********************************************************************/
void main()
{
   exer();
}
```

```
/************************************************************************/
/* Routine that asks the question, then depending on the user's answer      */
/* makes necessary explanations                                              */
/************************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  2");
  outtextxy(2*x,2*y,"Use the breadth first search algorithm to find a spanning tree");
  outtextxy(2*x,3*y,"(Start at A. If there is a choice of edges select edges according
                        to");
  outtextxy(2*x,4*y,"alphabetical order.)");
  pieslice(25*x,5*y,0,359,2);      /* A */
  pieslice(25*x,11*y,0,359,2);     /* B */
  pieslice(55*x,5*y,0,359,2);      /* C */
  pieslice(55*x,11*y,0,359,2);     /* D */
  pieslice(35*x,7*y,0,359,2);      /* E */
  pieslice(45*x,7*y,0,359,2);      /* F */
  pieslice(35*x,9*y,0,359,2);      /* G */
  pieslice(45*x,9*y,0,359,2);      /* H */
  outtextxy(25*x,9*y/2,"A");
  outtextxy(25*x,23*y/2,"B");
  outtextxy(55*x,9*y/2,"C");
  outtextxy(55*x,23*y/2,"D");
  outtextxy(33*x,7*y,"E");
  outtextxy(46*x,7*y,"F");
  outtextxy(33*x,9*y,"G");
  outtextxy(46*x,9*y,"H");
  moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
  lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
  lineto(45*x,7*y);  lineto(35*x,7*y); lineto(35*x,9*y);
```

```
 lineto(45*x,9*y);
 moveto(45*x,7*y);lineto(35*x,9*y);
 moveto(25*x,5*y);lineto(35*x,7*y);
while (in_the_exercise == 1) {
 outtextxy(15*x,14*y,"Choose one of the following, if you need :");
 outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
 outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
 outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
 outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
 outtextxy(15*x,19*y,"Enter your choice here --->");
 Ch = getch ();
 if(Ch==ESC) confirm_graph_exit();
   while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
     outtextxy(48*x,19*y,"    Please type a, b, c or d");
     Ch = getch ();
     if(Ch==ESC) confirm_graph_exit();
     if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
     setcolor(backcolor);
     bar(50*x,37*y/2,88*x,20*y);
     setcolor(forecolor);
     }
   }
     switch (Ch)          {
     case 'a': outtextxy(47*x,19*y,"a");
       outtextxy(52*x,19*y,"You want to see the algorithm ");
       outtextxy(52*x,20*y,"again. Press any key to continue.");
       Pause(30*x,24*y);
       setcolor(backcolor);
       bar(50*x,37*y/2,179*x/2,21*y);
       bar(2*x,13*y,179*x/2,49*y/2);
       setcolor(forecolor);
       show_alg();
       break;
    case 'b': outtextxy(47*x,19*y,"b");
       outtextxy(52*x,19*y,"You want to compare your solu-");
       outtextxy(52*x,20*y,"tion with ours. So press any  ");
```

```
      outtextxy(52*x,21*y,"key to see it.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,22*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      compare_solutions();
      break;
    case 'c': outtextxy(47*x,19*y,"c");
      outtextxy(52*x,19*y,"You want to see step by step");
      outtextxy(52*x,20*y,"solution. So press any key to ");
      outtextxy(52*x,21*y,"continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,22*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      step_solution();
      break;
    case 'd': outtextxy(47*x,19*y,"d");
      confirm_exit();
      break;
    default  : break;
   }
 }
  closegraph();
}
```

```
/***********************************************************************/
/* This routine gives breadth first search spanning tree algorithm          */
/***********************************************************************/
static void show_alg(void)
{
   outtextxy(15*x,12*y,"BREADTH  FIRST  SEARCH SPANNING TREE
                              ALGORITHM");
   outtextxy(2*x,13*y,"Step 1 (start with a vertex). Pick a vertex U and assign U the
                              label 0.");
   outtextxy(2*x,14*y,"Let  L = { x }, T = 0, and k = 0.");
   outtextxy(2*x,15*y,"Step 2 (L has n vertices). If L contains all the vertices of G,
                              then stop;");
   outtextxy(2*x,16*y,"the edges in T and the vertices in L form a spanning tree for
                              G.");
   outtextxy(2*x,17*y,"Step 3 (L has fewer than n vertices). If L does not contain all
                              the vertices");
   outtextxy(2*x,18*y,"of G, find the vertices not in L that are adjacent to the vertices
                              in L with");
   outtextxy(2*x,19*y,"largest label number k.If there are no such vertices,G has no
                              spanning tree.");
   outtextxy(2*x,20*y,"Otherwise, assign these newly found vertices the label k + 1
                              and put them in");
   outtextxy(2*x,21*y,"in L. For each new vertex with label k + 1, place in T one edge
                              connecting ");
   outtextxy(2*x,22*y,"this vertex to a vertex with label k. If there is more than one
                              such edge,");
   outtextxy(2*x,23*y,"choose one arbitrarily. Return to Step 2.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}
```

```
/***********************************************************************/
/* This routine gives the solution to the exercise to be compared.          */
/***********************************************************************/
static void compare_solutions(void)
{

   setcolor(backcolor);         /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   moveto(45*x,9*y);  lineto(55*x,11*y);  lineto(25*x,11*y);
   lineto(25*x,5*y);  lineto(55*x,5*y);
   moveto(35*x,9*y);  lineto(35*x,7*y);  lineto(45*x,7*y);
   moveto(25*x,5*y);  lineto(35*x,7*y);
   setcolor(forecolor);
   setlinestyle(3,0,3);
   outtextxy(27*x,9*y/2,"(0)");       /* A */
   outtextxy(22*x,11*y,"(1)");        /* B */
   outtextxy(56*x,5*y,"(1)");         /* C */
   outtextxy(36*x,13*y/2,"(1)");      /* E */
   outtextxy(56*x,11*y,"(2)");        /* D */
   outtextxy(43*x,13*y/2,"(2)");      /* F */
   outtextxy(36*x,19*y/2,"(2)");      /* G */
   outtextxy(47*x,9*y,"(3)");         /* H */
   moveto(45*x,9*y);  lineto(55*x,11*y);  lineto(25*x,11*y);
   lineto(25*x,5*y);  lineto(55*x,5*y);
   moveto(35*x,9*y);  lineto(35*x,7*y);  lineto(45*x,7*y);
   moveto(25*x,5*y);  lineto(35*x,7*y);
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,70*x,49*y/2);
   bar(2*x,17*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   pieslice(25*x,5*y,0,359,2);     /* A */   /* redraw the figure */
   pieslice(25*x,11*y,0,359,2);    /* B */
   pieslice(55*x,5*y,0,359,2);     /* C */
   pieslice(55*x,11*y,0,359,2);    /* D */
```

```
    pieslice(35*x,7*y,0,359,2);     /* E */
    pieslice(45*x,7*y,0,359,2);     /* F */
    pieslice(35*x,9*y,0,359,2);     /* G */
    pieslice(45*x,9*y,0,359,2);     /* H */
    outtextxy(25*x,9*y/2,"A");
    outtextxy(25*x,23*y/2,"B");
    outtextxy(55*x,9*y/2,"C");
    outtextxy(55*x,23*y/2,"D");
    outtextxy(33*x,7*y,"E");
    outtextxy(46*x,7*y,"F");
    outtextxy(33*x,9*y,"G");
    outtextxy(46*x,9*y,"H");
    moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
    lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
    lineto(45*x,7*y); lineto(35*x,7*y); lineto(35*x,9*y);
    lineto(45*x,9*y);
    moveto(45*x,7*y);lineto(35*x,9*y);
    moveto(25*x,5*y);lineto(35*x,7*y);
}
```

```
/*************************************************************************/
/* This routine gives the step by step solution to the exercise          */
/*************************************************************************/
static void step_solution(void)
{

  setcolor(backcolor);        /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  /**********************************************************************/
  outtextxy(64*x,5*y,"k");
  outtextxy(70*x,5*y,"L");
  outtextxy(75*x,5*y,"Label");
  outtextxy(86*x,5*y,"T");
  moveto(62*x,11*y/2); lineto(67*x,11*y/2);
  moveto(68*x,11*y/2); lineto(73*x,11*y/2);
  moveto(74*x,11*y/2); lineto(165*x/2,11*y/2);
  moveto(84*x,11*y/2); lineto(89*x,11*y/2);
  /**********************************************************************/
  outtextxy(70*x,6*y,"A");
  outtextxy(64*x,6*y,"0");
  outtextxy(75*x,6*y,"A <- 0");
  outtextxy(27*x,9*y/2,"(0)");
  /**********************************************************************/
  Pause(30*x,24*y);
  outtextxy(70*x,7*y,"B");
  outtextxy(70*x,8*y,"C");
  outtextxy(70*x,9*y,"E");
  outtextxy(75*x,7*y,"B <- 1");
  outtextxy(84*x,7*y,"(A,B)");
  outtextxy(22*x,11*y,"(1)");
  outtextxy(75*x,8*y,"C <- 1");
  outtextxy(84*x,8*y,"(A,C)");
  outtextxy(56*x,5*y,"(1)");
  outtextxy(75*x,9*y,"E <- 1");
  outtextxy(84*x,9*y,"(A,E)");
outtextxy(36*x,13*y/2,"(1)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(25*x,5*y);  lineto(25*x,11*y);
moveto(25*x,5*y);  lineto(55*x,5*y);
moveto(25*x,5*y);  lineto(35*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,5*y);  lineto(25*x,11*y);   /* add (A, B) to T */
moveto(25*x,5*y);  lineto(55*x,5*y);    /* add (A, C) to T */
moveto(25*x,5*y);  lineto(35*x,7*y);    /* add (A, E) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,7*y,"1");
/*****************************************************************/
Pause(30*x,24*y);
outtextxy(70*x,10*y,"D");
outtextxy(70*x,11*y,"F");
outtextxy(70*x,12*y,"G");
outtextxy(75*x,10*y,"D <- 2");
outtextxy(84*x,10*y,"(B,D)");
outtextxy(56*x,11*y,"(2)");
outtextxy(75*x,11*y,"F <- 2");
outtextxy(84*x,11*y,"(E,F)");
outtextxy(43*x,13*y/2,"(2)");
outtextxy(75*x,12*y,"G <- 2");
outtextxy(84*x,12*y,"(E,G)");
outtextxy(36*x,19*y/2,"(2)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(25*x,11*y);  lineto(55*x,11*y);
moveto(35*x,7*y);  lineto(45*x,7*y);
moveto(35*x,7*y);  lineto(35*x,9*y);
```

```
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,11*y);  lineto(55*x,11*y);  /* add (B, D) to T */
moveto(35*x,7*y);  lineto(45*x,7*y);   /* add (E, F) to T */
moveto(35*x,7*y);  lineto(35*x,9*y);   /* add (E, G) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(64*x,10*y,"2");
/*****************************************************************/
outtextxy(70*x,13*y,"H");
outtextxy(75*x,13*y,"H <- 3");
outtextxy(84*x,13*y,"(D,H)");
outtextxy(47*x,9*y,"(3)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(55*x,11*y);  lineto(45*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,9*y);  lineto(55*x,11*y);     /* add (D, H) to T */
setlinestyle(0,0,3);
outtextxy(64*x,13*y,"3");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(65*x,20*y,"We are done.");
/*****************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);          /* Clean the game field  again */
bar(3*x/2,17*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(25*x,5*y,0,359,2);      /* A */
```

```
 pieslice(25*x,11*y,0,359,2);      /* B */
 pieslice(55*x,5*y,0,359,2);       /* C */
 pieslice(55*x,11*y,0,359,2);      /* D */
 pieslice(35*x,7*y,0,359,2);       /* E */
 pieslice(45*x,7*y,0,359,2);       /* F */
 pieslice(35*x,9*y,0,359,2);       /* G */
 pieslice(45*x,9*y,0,359,2);       /* H */
/*********************************************************************/
 outtextxy(25*x,9*y/2,"A");
 outtextxy(25*x,23*y/2,"B");
 outtextxy(55*x,9*y/2,"C");
 outtextxy(55*x,23*y/2,"D");
 outtextxy(33*x,7*y,"E");
 outtextxy(46*x,7*y,"F");
 outtextxy(33*x,9*y,"G");
 outtextxy(46*x,9*y,"H");
/*********************************************************************/
 moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
 lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
 lineto(45*x,7*y);  lineto(35*x,7*y); lineto(35*x,9*y);
 lineto(45*x,9*y);
 moveto(45*x,7*y);lineto(35*x,9*y);
 moveto(25*x,5*y);lineto(35*x,7*y);
}
```

```
/**********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,23*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,22*y,179*x/2,49*y/2);
       setcolor(forecolor);
   }
    switch (ch)         {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;

    case 'n': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    case 'N': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    default : break;
    }
}
```

```
/* PROGRAM    : depth.c
   AUTHOR     : Atilla BAKAN
   DATE       : Mar. 16, 1990
   REVISED    : Mar. 16, 1990

   DESCRIPTION : This program contains the tutorial for depth first search
                 algoritm.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
```

```
  #define ff_attrib      attribute
#endif

#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void add_shadow    (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit    (int errnum);
static void initialize    (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void press_a_key   (int wrow);
static void Pageup        (void);
static void Pagedown      (void);
static void pre_help      (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);

/* Tutorial procedures              */
static void complexity    (void);
static void depth_first   (void);
static void ex_depth_1    (void);
static void theorem_4_8   (void);
static void proof_4_8     (void);
static void definition_4_3_1(void);
static void ex_depth_2    (void);
static void exercises     (void);
static void exer1         (void);
static void exer2         (void);
static void P1            (void);
static void P2          (void);
static void P3          (void);
static void P4          (void);
static void P5          (void);
static void P6          (void);
static void P7          (void);
static void P8          (void);
static void P9          (void);

/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];


/**********************************************************************/
/* error message table                                                */
/**********************************************************************/
static char *error_text[]= {
  NULL,  /* errnum =  0, no error    */
  NULL,  /* errnum == 1, windowing error */
  "Syntax:  CXLDEMO [-switches]\n\n"
     "\t -c = CGA snow elimination\n"
     "\t -b = BIOS screen writing\n"
     "\t -m = force monochrome text attributes",
  "Memory allocation error"
};


/**********************************************************************/
/* miscellaneous defines                                              */
/**********************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33
```

```
/**********************************************************************/
/* this function will add a shadow to the active window              */
/**********************************************************************/
static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}
/**********************************************************************/
/* this function pops open a window and confirms that the user really  */
/* wants to quit the demo.  If so, it terminates the demo program.     */
/**********************************************************************/
static void confirm_quit(void)
{
   struct _onkey_t *kblist;

   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
   add_shadow();
   wputs("\n Quit demo, are you sure? \033A\156Y\b");
   clearkeys();
   showcur();
   if(wgetchf("YN",'Y')=='Y') normal_exit();
   wclose();
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}

/**********************************************************************/
/* this function is called by the pull-down demo for a prompt         */
/**********************************************************************/
static void disp_sure_msg(void)
{
   wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
```

```
/*********************************************************************/
/* this function handles abnormal termination.  If it is passed an    */
/* error code of 1, then it is a windowing system error.  Otherwise   */
/* the error message is looked up in the error message table.        */
/*********************************************************************/
static void error_exit(int errnum)
{
  if(errnum) {
    printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
    exit(errnum);
  }
}

/*********************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems    */
/*********************************************************************/
static void initialize(void)
{
  /* initialize the CXL video system and save current screen info */
  videoinit();
  readcur(&crow,&ccol);
  if((savescrn=ssave())==NULL) error_exit(3);

  /* if mouse exists, turn on full mouse support */
  if(msinit()) {
    mssupport(MS_FULL);
    msgotoxy(12,49);
  }
  /* attach [Alt-X] to the confirm_quit() function */
  setonkey(0x2d00,confirm_quit,0);

  /* attach [Ctrl Pageup] to the Pageup() function */
  setonkey(0x8400,Pageup,0);

  /* attach [Ctrl Pagedown] to the Pagedown() function */
  setonkey(0x7600,Pagedown,0);
  /* initialize help system, help key = [F1] */
  whelpdef("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
           WHITE|_RED,RED|_LGREY,pre_help);
}


/*********************************************************************/
/* this function is called anytime to switch back to previous window.         */
/*********************************************************************/
static void Pageup(void)
{
  static WINDOW handle;

  handle = whandle();
  wactiv(handle - 1);
}


/*********************************************************************/
/* this function is called anytime to switch back to next window.             */
/*********************************************************************/
static void Pagedown(void)
{
  static WINDOW handle;

  handle = whandle();
  wactiv(handle + 1);
}


/*********************************************************************/
static void pre_help(void)
{
  add_shadow();
  setonkey(0x2d00,confirm_quit,0);
}
```

```
/*********************************************************************/
/* this function handles normal termination. The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.           */
/*********************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
/*********************************************************************/
/* this function displays a pause message then pauses for a keypress        */
/*********************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;

   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"           ");
   wprints(wrow,0,attr2,"         ");
}
/*********************************************************************/
/* This routine causes short dealys during execution                          */
/*********************************************************************/
static void short_delay(void)
{
   delay_(SHORT_DELAY);
}
```

```
/***********************************************************************/
/* this function is called by the pull-down menu demo anytime           */
/* the  selection bar moves on or off the [Q]uit menu items.            */
/***********************************************************************/
static void quit_window(void)
{
   static WINDOW handle=0;

   if(handle) {
      wactiv(handle);
      wclose();
      handle=0;
   }
   else {
      handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
      wputs(" Quit takes you back to the\n demo program's main menu.");
   }
}

/***********************************************************************/
/* shows the cursor again if it has been hidden                         */
/***********************************************************************/
static void restore_cursor(void)
{
   wtextattr(WHITE|_MAGENTA);
   showcur();
}

/***********************************************************************/
/* enlarges or shrinks the win    ws                                    */
/***********************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
```

```
/**********************************************************************/
/* moves the active window to a given screen coordinates             */
/**********************************************************************/
static void move_window(int nsrow,int nscol)
{
   if(wmove(nsrow,nscol)) error_exit(1);
   short_delay();
}
/**********************************************************************/
/* this routine  calls depth_first()  routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                  */
/**********************************************************************/
void P1()
{
   wcloseall();
   depth_first();
}
/**********************************************************************/
/* this routine  calls ex_depth_1 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P2()
{
   wcloseall();
   ex_depth_1();
}

/**********************************************************************/
/* this routine  calls ex_depth_2 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P3()
{
   wcloseall();
   ex_depth_2();
}
```

```
/************************************************************************/
/* this routine   calls theorem_4_8 routine whenever Pageup or             */
/* Pagedown keys are pressed.                                               */
/************************************************************************/
void P4()
{
   wcloseall();
   theorem_4_8();
}
/************************************************************************/
/* this routine   calls definition_4_3_1 routine whenever Pageup or        */
/* Pagedown keys are pressed.                                               */
/************************************************************************/
void P5()
{
   wcloseall();
   definition_4_3_1();
}
/************************************************************************/
/* this routine   calls complexity routine whenever Pageup or              */
/* Pagedown keys are pressed.                                               */
/************************************************************************/
void P6()
{
   wcloseall();
   complexity();
}
/************************************************************************/
/* this routine   calls exercises routine whenever Pageup or               */
/* Pagedown keys are pressed.                                               */
/************************************************************************/
void P7()
{
   wcloseall();
   exercises();
}
```

```
/*********************************************************************/
/* this routine  calls exer1  routine whenever Pageup or           */
/* Pagedown keys are pressed.                                       */
/*********************************************************************/
void P8()
{
  wcloseall();
  exer1();
}
```

```
/*********************************************************************/
/* this routine  calls exer2 routine whenever Pageup or            */
/* Pagedown keys are pressed.                                       */
/*********************************************************************/
void P9()
{
  wcloseall();
  exer2();
}
```

```
/*********************************************************************/
/* main routine which is  calling  minimal spanning tree tutorial    */
/*********************************************************************/
void main()
{
 initialize();
 depth_first();
}
```

```
/**********************************************************************/
/* This routine  calls definition, example and algorithm routines about      */
/* depth_first_search.                                                         */
/**********************************************************************/
static void depth_first(void)
{
   register int *scrn;

   if((scrn=ssave())==NULL) error_exit(3);
   cclrscrn(LGREY|_BLUE);
   /*******************************************************************/
   /* attach [Pagedown] to the ex_depth_1() function */
   setonkey(0x5100,P2,0);
   /*******************************************************************/
   if((w[1]=wopen(5,15,13,54,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
              error_exit(1);
   wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputsw(" Depth First Search algorithm is one of the algorithms that"
            " are used for finding spanning trees. In this algorithm we"
            " label the vertices with consecutive integers. The underlying"
            " idea of the algorithm is :");
   press_a_key(6);
   wslide(0,0);
  /*******************************************************************/
   if((w[2]=wopen(2,15,23,54,3,LCYAN|_BLACK,BLACK|_GREEN))==0)
             error_exit(1);
   wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputsw(" Find the vertex that should be labeled immediately after"
            " labeling vertex V, the first vertices to consider are the"
            " ones adjacent to V. If there is an unlabeled vertex W adjacent"
            " to vertex V, W is assigned the next label number, and the"
            " process of searching for the next vertex to label is begun"
            " with W. If V has no unlabeled adjacent vertices, we back up"
            " to the vertex that was labeled immediately before V and "
            " continue backing up, if necessary, until we reach a vertex"
            " having an unlabeled adjacent vertex U. Vertex U is then assigned"
           " the next label number, and the process of searching for the"
           " next vertex to label is begun with U.");
press_a_key(19);
wslide(0,40);
/*****************************************************************/
if((w[3]=wopen(5,15,10,65,3,LCYAN|_BLACK,BLACK|_RED))==0)
          error_exit(1);
wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputs("\n");
wputsw(" The formal specification of the depth_first search algorithm"
          " is as follows :");
press_a_key(3);
short_delay();
wcloseall();
/*****************************************************************/
if((w[1]=wopen(0,15,24,65,3,BLACK|_GREEN,BLACK|_CYAN))==0)
          error_exit(1);
wtitle("[Depth First Search Algorithm]",TCENTER,BLUE|_LGREY);
add_shadow();
wputs("\n");
wputsw(" It will label the vertices (1, ... ,n)");
wputs("\n");
wputsw(" Step 1 . Pick a vertex x.");
wputs("\n        L = { x } (list of vertices in the tree)");
wputs("       T = 0    (list of edges in the tree) ");
wputs("\n        x <- 1                       ");
wputs("\n        k <- 2 (counter)             \n");
wputsw(" Step 2 . Pick any vertex U not in L, such that U is adjacent"
         " to the vertex L with the highest label, say V. ");
wputs("\n        L = L U { U }                    ");
```

```
  wputs("\n       T = T U {U,V}                    ");
  wputs("\n       U <- k                      ");
  wputs("\n       k <- k + 1                  \n");
  wputs(" Step 3 . a) If all vertices are in L, stop\n");
  wputs("        b) If not all vertices are in L \n");
  wputsw("           1) If there exist a vertex adjacent not in L"
            " which is adjacent to a vertex in L, go to Step 2");
  wputs("\n");
  wputsw("           2) If no such vertex exists, stop and output"
            " that the graph is not connected.");
  press_a_key(22);
  ex_depth_1();
  srestore(scrn);
}


/************************************************************************/
/* An  example about a Depth First Search Algorithm implementation               */
/************************************************************************/
static void ex_depth_1 (void)
{
  /**********************************************************************/
  /* attach [Pageup] to the depth_first() function */
  setonkey(0x4900,P1,0);
  /**********************************************************************/
  /* attach [Pagedown] to the ex_depth_2() function */
  setonkey(0x5100,P3,0);
  /**********************************************************************/
  if((w[2]=wopen(5,15,10,65,3,LCYAN|_BLACK,BLACK|_RED))==0)
            error_exit(1);
  wtitle("[Depth First Search - Example_4_3_1]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n       We need to show an example !");
  press_a_key(3);
  short_delay();
```

```
   wcloseall();
   spawnl(P_WAIT,"examp431.exe",NULL);
   cclrscrn(LGREY|_BLUE);
   ex_depth_2();
}




/***********************************************************************/
/* Another example about a Depth First Search Algorithm implementation          */
/***********************************************************************/
static void ex_depth_2 (void)
{
   /********************************************************************/
   /* attach [Pageup] to the ex_depth_1() function */
   setonkey(0x4900,P2,0);
   /********************************************************************/
   /* attach [Pagedown] to the theorem_4_8() function */
   setonkey(0x5100,P4,0);
   /********************************************************************/
   if((w[3]=wopen(13,15,18,65,3,LCYAN|_BLACK,BLACK|_RED))==0)
            error_exit(1);
   wtitle("[Depth First Search - Example_4_3_2]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n     Now we will show you one more example.");
   wputs("\n        But a little bit complicated !");
   press_a_key(3);
   short_delay();
   /********************************************************************/
   wcloseall();
   spawnl(P_WAIT,"examp432.exe",NULL);
   cclrscrn(LGREY|_BLUE);
   theorem_4_8();
}
```

```
/*********************************************************************/
/* This routine gives the theorem_4_8 about the trees. Besides, if the user      */
/* wants, it gives the proof for this theorem.                                      */
/*********************************************************************/
static void theorem_4_8(void)
 {
  struct _onkey_t *kblist;

  /*****************************************************************/
  /* attach [Pageup] to the ex_depth_2() function */
  setonkey(0x4900,P3,0);
  /*****************************************************************/
  /* attach [Pagedown] to the definition_4_3_1() function */
  setonkey(0x5100,P5,0);
  /*****************************************************************/
  if((w[1]=wopen(5,15,10,65,3,WHITE|_CYAN,RED|_BLACK))==0) error_exit(1);
  wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n          We have a theorem for you! ");
  press_a_key(3);
  wclose();
  /*****************************************************************/
  if((w[1]=wopen(3,4,10,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
            error_exit(1);
  wtitle("[Depth First Search Algorithm - Theorem_4_8]",
           TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("Theorem_4_8\n");
  wputsw(" Let the depth first search algorithm be applied to a graph G");
  wputsw(" (a) The edges in T and the vertices in L form a tree.");
  wputs("\n");
  wputsw(" (b) Furthermore, if G is connected, this tree is a spanning"
            " tree.");
  press_a_key(5);
  /*******************************************************************/
  kblist=chgonkey(NULL); /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
  add_shadow();
  wputs("\n Do you want to see the proof? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') {
    wclose();
    proof_4_8();
  }
  else wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
  /*******************************************************************/
  wclose();
  definition_4_3_1();
}


/*******************************************************************/
/* This routine gives the proof to the theorem_4_8.                */
/*******************************************************************/
static void proof_4_8(void)
{
  /*******************************************************************/
  /* attach [Pageup] to the ex_depth_2() function */
  setonkey(0x4900,P3,0);
  /*******************************************************************/
  /* attach [Pagedown] to the definition_4_3_1() function */
  setonkey(0x5100,P5,0);
  /*******************************************************************/
  if((w[2]=wopen(11,4,23,71,3,LCYAN|_RED,WHITE|_GREEN))==0)
          error_exit(1);
```

```
   wtitle("[Theorem_4_8 - Proof]",TCENTER,_MAGENTA|WHITE);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n");
   wputsw(" (a) By the construction process of depth first search, the"
            " edges of T and the vertices in L form a connected graph"
            " In step 3 each time an edge is selected to be placed in T,"
           " one vertex is in L and the other is not in L. Thus, this"
           " selection does not create any cycles using other edges in T."
           " Consequently, at the end of the depth_first_search algorithm"
          " the graph formed by the edges in T and the vertices in L "
          " contains no cycles and is, therefore, a tree.");
   wputs("\nThe proof of part (b) is left as an exercise.");
   press_a_key(10);
   wclose();
}

/*************************************************************************/
/* This routine gives the definitions of the concepts related to the depth      */
/* first search algorithm.                                                       */
/*************************************************************************/
static void definition_4_3_1(void)
{
   /**********************************************************************/
   /* attach [Pageup] to the theorem_4_8() function */
   setonkey(0x4900,P4,0);
   /**********************************************************************/
   /* attach [Pagedown] to the complexity() function */
   setonkey(0x5100,P6,0);
   /**********************************************************************/
   if((w[1]=wopen(5,4,16,71,3,LCYAN|_RED,BROWN|_CYAN))==0) error_exit(1);
   wtitle("[Definition4_3_1]",TCENTER,_MAGENTA|WHITE);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n");
   wputsw(" Let's call the tree formed by the edges in T and the vertices"
            " in L (after depth first search algorithm applied) as simply T.");
   wputs("\n The tree T is called a depth first search tree.\n");
   wputsw(" The edges in T are called tree edges and the other edges are"
            " called back edges.");
   wputs("\n");
   wputsw(" The labeling of the vertices is called a depth first search"
            " numbering.");
   wputs("\n");
   wputsw(" Consider the graph in the example we showed to you :");
   press_a_key(9);
   wclose();
   spawnl(P_WAIT,"examp433.exe",NULL);
   cclrscrn(LGREY|_BLUE);
   complexity();
}

/*******************************************************************/
/* This routine tells about the efficiency of the depth first search alg.        */
/*******************************************************************/
static void complexity(void)
{
   /*****************************************************************/
   /* attach [Pageup] to the definition_4_3_1() function */
   setonkey(0x4900,P5,0);
   /*****************************************************************/
   /* attach [Pagedown] to the exercises() function */
   setonkey(0x5100,P7,0);
   /*****************************************************************/
   if((w[1]=wopen(2,15,17,65,3,LCYAN|_BLACK,WHITE|_MAGENTA))==0)
            error_exit(1);
   wtitle("[Depth First Search - Efficiency]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n");
   wputsw(" In order to analyze the complexity of the depth first search"
            " algorithm, we will regard labeling a vertex and using an edge"
```

```
          " as the elementary operations. For a graph with n vertices and"
          " e edges, each vertex is labeled at most once and each edge is"
          " used at most twice, once in going from a labeled vertex to an"
          " unlabeled vertex and once in backing up to a previously labeled"
          " vertex. Hence, there will be at most");
  wputs("\n         n + 2*e <= n + 2*1/2*n*(n -1)");
  wputsw(" operations, and thus this algorithm is of order at most n^2");
  press_a_key(13);
  wslide(0,0);
  /*********************************************************************/
  if((w[2]=wopen(5,15,14,65,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" Depth first search can be used in many other ways to solve"
          " problems involving graphs and directed graphs. But at "
          " this stage we will cover only this much. Because our "
          " intention is only to give you an idea about depth first"
          " search.");
  press_a_key(7);
  short_delay();
  wcloseall();
  exercises();
}
```

```
/************************************************************************/
/* This routine makes a small quiz about the depth first search.        */
/************************************************************************/
void exercises(void)
 {
   register int *screen;

   /*********************************************************************/
   /* attach [Pageup] to the complexity() function */
   setonkey(0x4900,P6,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exer1() function */
   setonkey(0x5100,P8,0);
   /*********************************************************************/
   if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
              error_exit(1);
   wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   add_shadow();
   wputs("\n");
   wputsw(" We have completed our presentation of this section. Are"
              " you ready for a pop quiz ? ");
   press_a_key(3);
   short_delay();
   wclose();
   if((screen=ssave())==NULL) error_exit(3); {
  exer1();
  /* if mouse exists, turn on full mouse support */
  if(msinit()) {
      mssupport(MS_FULL);
      msgotoxy(12,49);
      }
   }
 srestore(screen);
 }
```

```
/***********************************************************************/
/* Dummy function to call the actual exercise 4.3.1                   */
/***********************************************************************/
static void exer1(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the complexity() function          */
  setonkey(0x4900,P6,0);
  /*********************************************************************/
  /* attach [Pagedown] to the exer2() function */
  setonkey(0x5100,P9,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the first question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q431.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer2();
}
```

```
/************************************************************************/
/* Dummy function to call the actual exercise 4.3.2                     */
/************************************************************************/
static void exer2(void)
{
  /**********************************************************************/
  /* attach [Pageup] to the exer1() function          */
  setonkey(0x4900,P8,0);
  /**********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
            error_exit(1);
  wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the second question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q432.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  normal_exit();
}
```

```
/* PROGRAM   : examp431.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for depth first
                 search.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                     /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                      /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void ini _graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions      */
static void exer             (void);

/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*********************************************************************/
/* This function is used for including drivers to the executable code */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines            */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/*********************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)          {
   case 'y': closegraph();
           videoinit();
           exit(0);
           break;
   case 'Y': closegraph();
           videoinit();
           exit(0);
           break;
   case 'n': setcolor(backcolor);
           bar(4*x/3,23*y,30*x,97*y/4);
           bar(31*x,23*y,69*x,97*y/4);
           setcolor(forecolor);
           break;
   case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}



/*********************************************************************/
/* This function sets the text default values                        */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
```

```
/***********************************************************************/
/* main routine that calls exer routine                              */
/***********************************************************************/
void main()
{
  exer();
}

/***********************************************************************/
/* This routine illustrates  an implementation of depth first search    */
/***********************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE 4-3-1");
   /********************************************************************/
   pieslice(5*x,7*y,0,359,2);      /* A */
   pieslice(10*x,7*y,0,359,2);     /* B */
   pieslice(35*x,7*y,0,359,2);     /* G */
   pieslice(20*x,4*y,0,359,2);     /* D */
   pieslice(20*x,10*y,0,359,2);    /* E */
   pieslice(20*x,3*y/2,0,359,2);   /* C */
   pieslice(20*x,25*y/2,0,359,2);  /* F */
   outtextxy(3*x,7*y,"A");
   outtextxy(36*x,7*y,"G");
   outtextxy(20*x/2,15*y/2,"B");
   outtextxy(20*x,9*y/2,"D");
   outtextxy(21*x,3*y/2,"C");
   outtextxy(20*x,19*y/2,"E");
   outtextxy(21*x,25*y/2,"F");
   moveto(5*x,7*y); lineto(10*x,7*y); lineto(35*x,7*y);
   moveto(5*x,7*y); lineto(20*x,4*y);   lineto(10*x,7*y);
   moveto(5*x,7*y); lineto(20*x,3*y/2);  lineto(20*x,4*y);
moveto(5*x,7*y); lineto(20*x,10*y);  lineto(10*x,7*y);
moveto(5*x,7*y); lineto(20*x,25*y/2); lineto(20*x,10*y);
moveto(20*x,3*y/2);  lineto(35*x,7*y);
moveto(20*x,4*y);   lineto(35*x,7*y);
moveto(20*x,10*y);  lineto(35*x,7*y);
moveto(20*x,25*y/2); lineto(35*x,7*y);
/*******************************************************************/
outtextxy(45*x,2*y,"k");
outtextxy(48*x,2*y,"Vertex(V)");
outtextxy(62*x,3*y/2,"Adj");
outtextxy(59*x,2*y,"vertex(U)");
outtextxy(73*x,2*y,"Label");
outtextxy(82*x,2*y,"L");
outtextxy(87*x,2*y,"T");
moveto(44*x,5*y/2);  lineto(47*x,5*y/2);
moveto(48*x,5*y/2);  lineto(58*x,5*y/2);
moveto(59*x,5*y/2);  lineto(69*x,5*y/2);
moveto(70*x,5*y/2);  lineto(80*x,5*y/2);
moveto(81*x,5*y/2);  lineto(84*x,5*y/2);
moveto(85*x,5*y/2);  lineto(90*x,5*y/2);
outtextxy(2*x,14*y,"THE WAY WE APPLIED THE ALGORITHM");
moveto(3*x/2,29*y/2);  lineto(43*x,29*y/2);
/*******************************************************************/
outtextxy(2*x,15*y,". Initially pick (arbitrarily) A and");
outtextxy(2*x,16*y,"  put it in vertex(V).");
Pause(30*x,24*y);
outtextxy(52*x,3*y,"A");
/*******************************************************************/
outtextxy(2*x,17*y,". Label A with 1 (As you see, we will");
outtextxy(2*x,18*y,"  show the label in paranthesis near");
outtextxy(2*x,19*y,"  the vertex.");
Pause(30*x,24*y);
outtextxy(72*x,3*y,"A <- 1");
outtextxy(2*x,13*y/2,"(1)");
/*******************************************************************/
outtextxy(2*x,20*y,". Put A in L.");
```

```
Pause(30*x,24*y);
outtextxy(82*x,3*y,"A");
/*****************************************************************/
outtextxy(2*x,21*y,". Increment k.");
Pause(30*x,24*y);
outtextxy(45*x,4*y,"2");
/*****************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,50*x,24*y);
bar(3*x/2,23*y,179*x/2,97*y/4);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Now pick F since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to A (currently largest labeled).");
Pause(30*x,24*y);
outtextxy(52*x,4*y,"A");
outtextxy(63*x,4*y,"F");
/*****************************************************************/
outtextxy(2*x,17*y,". Label F with 2.");
Pause(30*x,24*y);
outtextxy(72*x,4*y,"F <- 2");
outtextxy(19*x,13*y,"(2)");
/*****************************************************************/
outtextxy(2*x,18*y,". Put F in L and put (A,F) in T.");
Pause(30*x.24*y);
outtextxy(82*x,4*y,"F");
outtextxy(85*x,4*y,"(A,F)");
setcolor(backcolor);
moveto(5*x,7*y);  lineto(20*x,25*y/2);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(5*x,7*y);  lineto(20*x,25*y/2);  /* add (A,F) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,19*y,". Increment k.");
```

```
Pause(30*x,24*y);
outtextxy(45*x,5*y,"3");
/*********************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
bar(3*x/2,23*y,179*x/2,97*y/4);
setcolor(forecolor);
/*********************************************************************/
outtextxy(2*x,15*y,". Now pick G since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to the (currently) largest");
outtextxy(2*x,17*y,"  labeled vertex F.");
Pause(30*x,24*y);
outtextxy(52*x,5*y,"F");
outtextxy(63*x,5*y,"G");
/*********************************************************************/
outtextxy(2*x,18*y,". Label G with 3.");
Pause(30*x,24*y);
outtextxy(72*x,5*y,"G <- 3");
outtextxy(37*x,7*y,"(3)");
/*********************************************************************/
outtextxy(2*x,19*y,". Put F in L and put (F,G) in T.");
Pause(30*x,24*y);
outtextxy(82*x,5*y,"G");
outtextxy(85*x,5*y,"(F,G)");
setcolor(backcolor);
moveto(20*x,25*y/2); lineto(35*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,25*y/2); lineto(35*x,7*y); /* add (F,G) to T */
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(30*x,24*y);
outtextxy(45*x,6*y,"4");
/*********************************************************************/
```

```
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
bar(3*x/2,23*y,179*x/2,97*y/4);
setcolor(forecolor);
/***********************************************************************/
outtextxy(2*x,15*y,". Pick C since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to the (currently) largest");
outtextxy(2*x,17*y,"  labeled vertex G.");
Pause(30*x,24*y);
outtextxy(52*x,6*y,"G");
outtextxy(63*x,6*y,"C");
/***********************************************************************/
outtextxy(2*x,18*y,". Label C with 4.");
Pause(30*x,24*y);
outtextxy(72*x,6*y,"C <- 4");
outtextxy(22*x,3*y/2,"(4)");
/***********************************************************************/
outtextxy(2*x,19*y,". Put C in L and put (G,C) in T.");
Pause(30*x,24*y);
outtextxy(82*x,6*y,"C");
outtextxy(85*x,6*y,"(G,C)");
setcolor(backcolor);
moveto(20*x,3*y/2);  lineto(35*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,3*y/2);  lineto(35*x,7*y);  /* add (G,C) to T */
setlinestyle(0,0,3);
/***********************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(30*x,24*y);
outtextxy(45*x,7*y,"5");
/***********************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
bar(3*x/2,23*y,179*x/2,97*y/4);
setcolor(forecolor);
/*********************************************************************/
outtextxy(2*x,15*y,". Pick D since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to the (currently) largest");
outtextxy(2*x,17*y,"  labeled vertex C.");
Pause(30*x,24*y);
outtextxy(52*x,7*y,"C");
outtextxy(63*x,7*y,"D");
/*********************************************************************/
outtextxy(2*x,18*y,". Label D with 5.");
Pause(30*x,24*y);
outtextxy(72*x,7*y,"D <- 5");
outtextxy(19*x,5*y,"(5)");
/*********************************************************************/
outtextxy(2*x,19*y,". Put D in L and put (C,D) in T.");
Pause(30*x,24*y);
outtextxy(82*x,7*y,"D");
outtextxy(85*x,7*y,"(C,D)");
setcolor(backcolor);
moveto(20*x,3*y/2);  lineto(20*x,4*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,3*y/2);  lineto(20*x,4*y);  /* add (C,D) to T */
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(30*x,24*y);
outtextxy(45*x,8*y,"6");
/*********************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
bar(3*x/2,23*y,179*x/2,97*y/4);
setcolor(forecolor);
```

```
/*********************************************************************/
outtextxy(2*x,15*y,". Pick B since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to the (currently) largest");
outtextxy(2*x,17*y,"  labeled vertex D.");
Pause(30*x,24*y);
outtextxy(52*x,8*y,"D");
outtextxy(63*x,8*y,"B");
/*********************************************************************/
outtextxy(2*x,18*y,". Label B with 6.");
Pause(30*x,24*y);
outtextxy(72*x,8*y,"B <- 6");
outtextxy(8*x,27*y/4,"(6)");
/*********************************************************************/
outtextxy(2*x,19*y,". Put B in L and put (D,B) in T.");
Pause(30*x,24*y);
outtextxy(82*x,8*y,"B");
outtextxy(85*x,8*y,"(D,B)");
setcolor(backcolor);
moveto(10*x,7*y);  lineto(20*x,4*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(10*x,7*y);  lineto(20*x,4*y);  /* add (D,B) to T */
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(30*x,24*y);
outtextxy(45*x,9*y,"7");
/*********************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
bar(3*x/2,23*y,179*x/2,97*y/4);
setcolor(forecolor);
/*********************************************************************/
outtextxy(2*x,15*y,". Pick E since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to  the (currently) largest");
```

```
    outtextxy(2*x,17*y," labeled vertex B.");
    Pause(30*x,24*y);
    outtextxy(52*x,9*y,"B");
    outtextxy(63*x,9*y,"E");
    /*****************************************************************/
    outtextxy(2*x,18*y,". Label E with 7.");
    Pause(30*x,24*y);
    outtextxy(72*x,9*y,"E <- 7");
    outtextxy(19*x,9*y,"(7)");
    /*****************************************************************/
    outtextxy(2*x,19*y,". Put E in L and put (B,E) in T.");
    Pause(30*x,24*y);
    outtextxy(82*x,9*y,"E");
    outtextxy(85*x,9*y,"(B,E)");
    setcolor(backcolor);
    moveto(10*x,7*y); lineto(20*x,10*y);
    setcolor(forecolor);
    setlinestyle(3,0,3);
    moveto(10*x,7*y); lineto(20*x,10*y); /* add (B,E) to T */
    setlinestyle(0,0,3);
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(3*x/2,59*y/4,55*x,24*y);
    bar(3*x/2,23*y,179*x/2,97*y/4);
    setcolor(forecolor);
    /*****************************************************************/
    outtextxy(2*x,15*y,". At this stage as you see all vertices are");
    outtextxy(2*x,16*y," in L. This means we are done. The graph");
    outtextxy(2*x,17*y," with dashed lines is the spanning tree");
    outtextxy(2*x,18*y," of underlying graph G.");
    /*****************************************************************/
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : examp432.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for implementation of
                 the depth first search.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                   /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)           /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions     */
static void exer            (void);

/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*********************************************************************/
/* This function is used for including drivers to the executable code */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/****************************************************************************/
/* This fuction initializes the necessary graphical routines               */
/****************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
   forecolor = WHITE;
 }
```

```
/*********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
   }
   switch (ch)         {
    case 'y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);      /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                           */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen               */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                 */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates an implementation of the depth first search.         */
/*********************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE 4-3-2");
   /******************************************************************/
   pieslice(10*x,4*y,0,359,2);     /* A */
   pieslice(20*x,4*y,0,359,2);     /* B */
   pieslice(30*x,4*y,0,359,2);     /* C */
   pieslice(40*x,4*y,0,359,2);     /* D */
   pieslice(10*x,7*y,0,359,2);     /* E */
   pieslice(20*x,7*y,0,359,2);     /* F */
   pieslice(30*x,7*y,0,359,2);     /* G */
   pieslice(40*x,7*y,0,359,2);     /* H */
   pieslice(20*x,10*y,0,359,2);     /* I */
   pieslice'30*x,10*y,0,359,2);     /* J */
   outtextxy(10*x,7*y/2,"A");
   outtextxy(20*x,7*y/2,"B");
   outtextxy(30*x,7*y/2,"C");
   outtextxy(40*x,7*y/2,"D");
   outtextxy(8*x,7*y,"E");
   outtextxy(18*x,15*y/2,"F");
   outtextxy(32*x,15*y/2,"G");
   outtextxy(42*x,7*y,"H");
   outtextxy(20*x,21*y/2,"I");
   outtextxy(30*x,21*y/2,"J");
   moveto(10*x,4*y); lineto(20*x,4*y); lineto(20*x,7*y);
   lineto(10*x,7*y); lineto(10*x,4*y);
   moveto(30*x,7*y); lineto(30*x,10*y);lineto(20*x,10*y);
   lineto(20*x,7*y); lineto(30*x,7*y);
   moveto(20*x,10*y);lineto(30*x,7*y);
moveto(30*x,7*y); lineto(30*x,4*y); lineto(40*x,4*y);
lineto(40*x,7*y);lineto(30*x,7*y);
moveto(30*x,4*y);lineto(40*x,7*y);
/*******************************************************************/
outtextxy(45*x,2*y,"k");
outtextxy(48*x,2*y,"Vertex(V)");
outtextxy(62*x,3*y/2,"Adj");
outtextxy(59*x,2*y,"vertex(U)");
outtextxy(73*x,2*y,"Label");
outtextxy(82*x,2*y,"L");
outtextxy(87*x,2*y,"T");
moveto(44*x,5*y/2);  lineto(47*x,5*y/2);
moveto(48*x,5*y/2);  lineto(58*x,5*y/2);
moveto(59*x,5*y/2);  lineto(69*x,5*y/2);
moveto(70*x,5*y/2);  lineto(80*x,5*y/2);
moveto(81*x,5*y/2);  lineto(84*x,5*y/2);
moveto(85*x,5*y/2);  lineto(90*x,5*y/2);
outtextxy(2*x,14*y,"THE WAY WE APPLIED THE ALGORITHM");
moveto(3*x/2,29*y/2);  lineto(43*x,29*y/2);
/*******************************************************************/
outtextxy(2*x,15*y,". Initially pick (arbitrarily) A and");
outtextxy(2*x,16*y,"  put it in vertex(V).");
Pause(7*x,24*y);
outtextxy(52*x,3*y,"A");
/*******************************************************************/
outtextxy(2*x,17*y,". Label A with 1 (As you see,we will");
outtextxy(2*x,18*y,"  show the label in paranthesis near");
outtextxy(2*x,19*y,"  the vertex.");
Pause(7*x,24*y);
outtextxy(72*x,3*y,"A <- 1");
outtextxy(6*x,4*y,"(1)");
/*******************************************************************/
outtextxy(2*x,20*y,". Put A in L.");
Pause(7*x,24*y);
outtextxy(82*x,3*y,"A");
/*******************************************************************/
```

```
outtextxy(2*x,21*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,4*y,"2");
/*****************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Now pick B since it is unlabeled ");
outtextxy(2*x,16*y,"  and adjacent to A (currently larg-");
outtextxy(2*x,17*y,"  est labeled).");
Pause(7*x,24*y);
outtextxy(52*x,4*y,"A");
outtextxy(63*x,4*y,"B");
/*****************************************************************/
outtextxy(2*x,18*y,". Label B with 2.");
Pause(7*x,24*y);
outtextxy(72*x,4*y,"B <- 2");
outtextxy(22*x,4*y,"(2)");
/*****************************************************************/
outtextxy(2*x,19*y,". Put B in L and put (A,B) in T.");
Pause(7*x,24*y);
outtextxy(82*x,4*y,"B");
outtextxy(85*x,4*y,"(A,B)");
setcolor(backcolor);
moveto(10*x,4*y); lineto(20*x,4*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(10*x,4*y); lineto(20*x,4*y); /* add (A,B) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,5*y,"3");
/*****************************************************************/
```

```
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(2*x,15*y,". Now pick F since it is unlabeled ");
outtextxy(2*x,16*y," and adjacent to the (currently)");
outtextxy(2*x,17*y," largest labeled vertex G.");
Pause(7*x,24*y);
outtextxy(52*x,5*y,"B");
outtextxy(63*x,5*y,"F");
/*******************************************************************/
outtextxy(2*x,18*y,". Label F with 3."),
Pause(7*x,24*y);
outtextxy(72*x,5*y,"F <- 3");
outtextxy(21*x,27*y/4,"(3)");
/*******************************************************************/
outtextxy(2*x,19*y,". Put F in L and put (B,F) in T.");
Pause(7*x,24*y);
outtextxy(82*x,5*y,"F");
outtextxy(85*x,5*y,"(B,F)");
setcolor(backcolor);
moveto(20*x,4*y); lineto(20*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,4*y); lineto(20*x,7*y); /* add (B,F) to T */
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,6*y,"4");
/*******************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,24*y);
setcolor(forecolor);
```

```
/*****************************************************************/
outtextxy(2*x,15*y,". Pick G since it is unlabeled and");
outtextxy(2*x,16*y," adjacent to the (currently) largest");
outtextxy(2*x,17*y," labeled vertex F.");
Pause(7*x,24*y);
outtextxy(52*x,6*y,"F");
outtextxy(63*x,6*y,"G");
/*****************************************************************/
outtextxy(2*x,18*y,". Label G with 4.");
Pause(7*x,24*y);
outtextxy(72*x,6*y,"G <- 4");
outtextxy(26*x,27*y/4,"(4)");
/*****************************************************************/
outtextxy(2*x,19*y,". Put G in L and put (F,G) in T.");
Pause(7*x,24*y);
outtextxy(82*x,6*y,"G");
outtextxy(85*x,6*y,"(F,G)");
setcolor(backcolor);
moveto(20*x,7*y); lineto(30*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,7*y); lineto(30*x,7*y); /* add (F,G) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,7*y,"5");
/*****************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Pick C since it is unlabeled and");
outtextxy(2*x,16*y," adjacent to the (currently) largest");
outtextxy(2*x,17*y," labeled vertex G.");
```

```
Pause(7*x,24*y);
outtextxy(52*x,7*y,"G");
outtextxy(63*x,7*y,"C");
/**********************************************************************/
outtextxy(2*x,18*y,". Label C with 5.");
Pause(7*x,24*y);
outtextxy(72*x,7*y,"C <- 5");
outtextxy(26*x,4*y,"(5)");
/**********************************************************************/
outtextxy(2*x,19*y,". Put C in L and put (G,C) in T.");
Pause(7*x,24*y);
outtextxy(82*x,7*y,"C");
outtextxy(85*x,7*y,"(G,C)");
setcolor(backcolor);
moveto(30*x,7*y); lineto(30*x,4*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(30*x,7*y); lineto(30*x,4*y); /* add (G,C) to T */
setlinestyle(0,0,3);
/**********************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,8*y,"6");
/**********************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
outtextxy(2*x,15*y,". Pick D since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to the (currently) largest");
outtextxy(2*x,17*y,"  labeled vertex C.");
Pause(7*x,24*y);
outtextxy(52*x,8*y,"C");
outtextxy(63*x,8*y,"D");
/**********************************************************************/
```

```
outtextxy(2*x,18*y,". Label D with 6.");
Pause(7*x,24*y);
outtextxy(72*x,8*y,"D <- 6");
outtextxy(41*x,4*y,"(6)");
/*********************************************************************/
outtextxy(2*x,19*y,". Put D in L and put (C,D) in T.");
Pause(7*x,24*y);
outtextxy(82*x,8*y,"D");
outtextxy(85*x,8*y,"(C,D)");
setcolor(backcolor);
moveto(30*x,4*y);  lineto(40*x,4*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(30*x,4*y);  lineto(40*x,4*y);  /* add (C,D) to T */
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,9*y,"7");
/*********************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,24*y);
setcolor(forecolor);
/*********************************************************************/
outtextxy(2*x,15*y,". Pick H since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to the (currently) largest");
outtextxy(2*x,17*y,"  labeled vertex D.");
Pause(7*x,24*y);
outtextxy(52*x,9*y,"D");
outtextxy(63*x,9*y,"H");
/*********************************************************************/
outtextxy(2*x,18*y,". Label H with 7.");
Pause(7*x,24*y);
outtextxy(72*x,9*y,"H <- 7");
outtextxy(38*x,15*y/2,"(7)");
```

```
/*******************************************************************/
outtextxy(2*x,19*y,". Put H in L and put (D,H) in T.");
Pause(7*x,24*y);
outtextxy(82*x,9*y,"H");
outtextxy(85*x,9*y,"(D,H)");
setcolor(backcolor);
moveto(40*x,4*y); lineto(40*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(40*x,4*y); lineto(40*x,7*y); /* add (D,H) to T */
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,10*y,"8");
/*******************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(2*x,15*y,". As you see there are still vertices");
outtextxy(2*x,16*y,"  with labels that have adj. vertices");
outtextxy(2*x,17*y,"  without labels. But, this time it's" );
outtextxy(2*x,18*y,"  not the vertex H with the largest ");
outtextxy(2*x,19*y,"  label which has such adj. vertices.");
outtextxy(2*x,20*y,"  Thus we back up until we find a la-");
outtextxy(2*x,21*y,"  beled vertex which has an unlabeled");
outtextxy(2*x,22*y,"  adjacent vertex. (i.e. G)");
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,24*y);
setcolor(forecolor);
outtextxy(52*x,10*y,"H");
outtextxy(63*x,10*y,"-");
outtextxy(52*x,11*y,"D");
outtextxy(63*x,11*y,"-");
outtextxy(52*x,12*y,"C");
outtextxy(63*x,12*y,"-");
/*****************************************************************/
outtextxy(2*x,15*y,". At this stage we choose J, since it");
outtextxy(2*x,16*y,"  is not labeled.");
Pause(7*x,24*y);
outtextxy(52*x,13*y,"G");
outtextxy(63*x,13*y,"J");
/*****************************************************************/
outtextxy(2*x,18*y,". Label J with 6.");
Pause(7*x,24*y);
outtextxy(72*x,13*y,"J <- 8");
outtextxy(32*x,10*y,"(8)");
/*****************************************************************/
outtextxy(2*x,19*y,". Put J in L and put (G,J) in T.");
Pause(7*x,24*y);
outtextxy(82*x,13*y,"J");
outtextxy(85*x,13*y,"(G,J)");
setcolor(backcolor);
moveto(30*x,7*y);  lineto(30*x,10*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(30*x,7*y);  lineto(30*x,10*y);  /* add (G,J) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,14*y,"9");
/*****************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Pick I since it is unlabeled and");
outtextxy(2*x,16*y," adjacent to the (currently) largest");
outtextxy(2*x,17*y," labeled vertex J.");
Pause(7*x,24*y);
outtextxy(52*x,14*y,"J");
outtextxy(63*x,14*y,"I");
/*****************************************************************/
outtextxy(2*x,18*y,". Label I with 9.");
Pause(7*x,24*y);
outtextxy(72*x,14*y,"I <- 9");
outtextxy(16*x,10*y,"(9)");
/*****************************************************************/
outtextxy(2*x,19*y,". Put I in L and put (J,I) in T.");
Pause(7*x,24*y);
outtextxy(82*x,14*y,"I");
outtextxy(85*x,14*y,"(J,I)");
setcolor(backcolor);
moveto(20*x,10*y); lineto(30*x,10*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,10*y); lineto(30*x,10*y); /* add (J,I) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,20*y,". Increment k.");
Pause(7*x,24*y);
outtextxy(45*x,15*y,"10");
/*****************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". As you see there are still vertices");
outtextxy(2*x,16*y," with labels that have adj. vertices");
outtextxy(2*x,17*y," without labels. But these are not ");
outtextxy(2*x,18*y," adj, to the vertex I with the larg-");
outtextxy(2*x,19*y," est label 9. Thus again, we back up");
outtextxy(2*x,20*y,"  to the vertex with label 5, and ");
outtextxy(2*x,21*y,"  find that it is adj. to a vertex ");
outtextxy(2*x,22*y,"  without label.");
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,44*x,49*y/2);
setcolor(forecolor);
outtextxy(52*x,16*y,"I");
outtextxy(63*x,16*y,"-");
outtextxy(52*x,17*y,"J");
outtextxy(63*x,17*y,"-");
outtextxy(52*x,18*y,"H");
outtextxy(63*x,18*y,"-");
outtextxy(52*x,19*y,"D");
outtextxy(63*x,19*y,"-");
outtextxy(52*x,20*y,"C");
outtextxy(63*x,20*y,"-");
outtextxy(52*x,21*y,"G");
outtextxy(63*x,21*y,"-");
/*********************************************************************/
outtextxy(2*x,15*y,". At this stage we choose E, since");
outtextxy(2*x,16*y,"  it is not labeled.");
Pause(7*x,24*y);
outtextxy(52*x,22*y,"F");
outtextxy(63*x,22*y,"E");
/*********************************************************************/
outtextxy(2*x,18*y,". Label E with 10.");
Pause(7*x,24*y);
outtextxy(72*x,22*y,"E <- 10");
outtextxy(8*x,15*y/2,"(10)");
/*********************************************************************/
outtextxy(2*x,19*y,". Put E in L and put (F,E) in T.");
Pause(7*x,24*y);
outtextxy(82*x,22*y,"E");
outtextxy(85*x,22*y,"(F,E)");
setcolor(backcolor);
```

```
    moveto(20*x,7*y); lineto(10*x,7*y);
    setcolor(forecolor);
    setlinestyle(3,0,3);
    moveto(20*x,7*y); lineto(10*x,7*y); /* add (F,E) to T */
    setlinestyle(0,0,3);
    /*******************************************************************/
    Pause(7*x,24*y);
    setcolor(backcolor);
    bar(3*x/2,59*y/4,44*x,49*y/2);
    setcolor(forecolor);
    /*******************************************************************/
    outtextxy(2*x,15*y,". At this stage as you see all ver-");
    outtextxy(2*x,16*y,"  tices are in L. This means we are");
    outtextxy(2*x,17*y,"  done. The graph with dashed lines");
    outtextxy(2*x,18*y,"  is the spanning tree of underlying");
    outtextxy(2*x,19*y,"  graph G.");
    /*******************************************************************/
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : examp433.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for a depth first search tree.



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/



/* header files */
#include <graphics.h>
#include "cxldef.h"

#if defined(__TURBOC__)                /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
   #define bioskey(a)     _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)    _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name        name
#elif defined(__ZTC__)                 /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name        name
   #define ff_attrib      attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer           (void);

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;



/**********************************************************************/
/* This function is used for including drivers to the executable code */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/****************************************************************************/
/* This fuction initializes the necessary graphical routines                */
/****************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**************************************************************************/
  settext();
  /**************************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      }
  forecolor = WHITE;
 }
```

```
/***********************************************************************/
/* This function sets the text default values                         */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) {
      closegraph();
      videoinit();
      exit(0);
    }
  }



/***********************************************************************/
/* main routine which calls exer routine                               */
/***********************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates a depth first search tree.              */
/*********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-3-3");
  /*****************************************************************/
  pieslice(5*x,7*y,0,359,2);      /* A */
  pieslice(10*x,7*y,0,359,2);     /* B */
  pieslice(35*x,7*y,0,359,2);     /* G */
  pieslice(20*x,4*y,0,359,2);     /* D */
  pieslice(20*x,10*y,0,359,2);    /* E */
  pieslice(20*x,3*y/2,0,359,2);   /* C */
  pieslice(20*x,25*y/2,0,359,2);  /* F */
  outtextxy(3*x,7*y,"A");
  outtextxy(36*x,7*y,"G");
  outtextxy(20*x/2,15*y/2,"B");
  outtextxy(20*x,9*y/2,"D");
  outtextxy(21*x,3*y/2,"C");
  outtextxy(20*x,19*y/2,"E");
  outtextxy(21*x,25*y/2,"F");
  moveto(5*x,7*y); lineto(10*x,7*y); lineto(35*x,7*y);
  moveto(5*x,7*y); lineto(20*x,4*y);   lineto(10*x,7*y);
  moveto(5*x,7*y); lineto(20*x,3*y/2);  lineto(20*x,4*y);
  moveto(5*x,7*y); lineto(20*x,10*y);  lineto(10*x,7*y);
  moveto(5*x,7*y); lineto(20*x,25*y/2); lineto(20*x,10*y);
  moveto(20*x,3*y/2);  lineto(35*x,7*y);
  moveto(20*x,4*y);    lineto(35*x,7*y);
  moveto(20*x,10*y);   lineto(35*x,7*y);
  moveto(20*x,25*y/2); lineto(35*x,7*y);
  /*****************************************************************/
  outtextxy(44*x,3*y,"EXPLANATIONS");
```

```
moveto(43*x,7*y/2); lineto(89*x,7*y/2);
/*********************************************************************/
outtextxy(2*x,13*y/2,"(1)");
outtextxy(19*x,13*y,"(2)");
outtextxy(37*x,7*y,"(3)");
outtextxy(22*x,3*y/2,"(4)");
outtextxy(19*x,5*y,"(5)");
outtextxy(8*x,27*y/4,"(6)");
outtextxy(19*x,9*y,"(7)");
setcolor(backcolor);
moveto(5*x,7*y); lineto(20*x,25*y/2);
moveto(35*x,7*y); lineto(20*x,25*y/2);
moveto(35*x,7*y); lineto(20*x,3*y/2);
moveto(20*x,4*y); lineto(20*x,3*y/2);
moveto(20*x,4*y); lineto(10*x,7*y);
moveto(20*x,10*y); lineto(10*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(5*x,7*y); lineto(20*x,25*y/2);
moveto(35*x,7*y); lineto(20*x,25*y/2);
moveto(35*x,7*y); lineto(20*x,3*y/2);
moveto(20*x,4*y); lineto(20*x,3*y/2);
moveto(20*x,4*y); lineto(10*x,7*y);
moveto(20*x,10*y); lineto(10*x,7*y);
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(43*x,4*y,"According to our definition, the tree");
outtextxy(43*x,5*y,"formed by dashed lines in the graph is");
outtextxy(43*x,6*y,"called depth first search tree.");
outtextxy(43*x,7*y,"The edges, such as (A,F), (C,D) are");
outtextxy(43*x,8*y,"called tree edges.");
outtextxy(43*x,9*y,"The edges, such as (A,B), (A,G),");
outtextxy(43*x,10*y,"are called back edges.");
outtextxy(43*x,11*y,"The labels (in paranthesis) of each ");
outtextxy(43*x,12*y,"vertex are called depth first search");
outtextxy(43*x,13*y,"numbers.");
```

```
  outtextxy(43*x,14*y,"Here we would like to take your atten-");
  outtextxy(43*x,15*y,"tion to the point that the designation");
  outtextxy(43*x,16*y,"of edges as tree and back edges as ");
  outtextxy(43*x,17*y,"well as the depth first search num-");
  outtextxy(43*x,18*y,"bering depends upon the choices made");
  outtextxy(43*x,19*y,"during implementation of the algorithm.");
        /*****************************************************************_
********/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
}
```

```
/* PROGRAM  : q431.c
   AUTHOR   : Atilla BAKAN
   DATE     : Mar. 22, 1990
   REVISED  : Apr. 17, 1990

   DESCRIPTION : This program contains the first exercise about the
                 depth first search for spanning trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)           /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/***********************************************************************/
/* miscellaneous global variables                                      */
/***********************************************************************/
 int in_the_exercise = 1;


/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/***********************************************************************/
/* This function is used for including drivers to the executable code          */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/***********************************************************************/
/* This fuction initializes the necessary graphical routines                    */
/***********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*********************************************************************/
  settext();
  /*********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
       }
   forecolor = WHITE;
 }

/**********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(32*x,24*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(31*x,23*y,69*x,97*y/4);
       setcolor(quitcolor);
    }
    switch (ch)         {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
      case 'Y': closegraph();
            videoinit();
            e..it(0);
            break;
      case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}



/*********************************************************************/
/* This function sets the text default values                        */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }


/**********************************************************************/
/* main routine that calls exer routine                               */
/**********************************************************************/
void main()
{
  exer();
}
```

```
/***********************************************************************/
/* This routine  asks the question, then depending on the user's answer        */
/* makes necessary explanations                                                  */
/***********************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  1");
  /**********************************************************************/
  outtextxy(2*x,2*y,"Use the depth first search algorithm to find a minimal spanning
                      tree.");
  outtextxy(2*x,3*y,"(Start at A. If there is a choice of edges select edges according
                      to");
  outtextxy(2*x,4*y,"alphabetical order.)");
  /**********************************************************************/
  pieslice(10*x,5*y,0,359,2);      /* A */
  pieslice(20*x,5*y,0,359,2);      /* B */
  pieslice(10*x,13*y/2,0,359,2);   /* C */
  pieslice(20*x,13*y/2,0,359,2);   /* D */
  pieslice(30*x,13*y/2,0,359,2);   /* E */
  pieslice(20*x,8*y,0,359,2);      /* F */
  pieslice(30*x,8*y,0,359,2);      /* G */
  pieslice(40*x,8*y,0,359,2);      /* H */
  pieslice(30*x,19*y/2,0,359,2);   /* I */
  pieslice(40*x,19*y/2,0,359,2);   /* J */
  pieslice(50*x,19*y/2,0,359,2);   /* K */
  pieslice(40*x,11*y,0,359,2);     /* L */
  pieslice(50*x,11*y,0,359,2);     /* M */
  /**********************************************************************/
  outtextxy(10*x,9*y/2,"A");
  outtextxy(20*x,9*y/2,"B");
outtextxy(8*x,13*y/2,"C");
outtextxy(21*x,25*y/4,"D");
outtextxy(31*x,13*y/2,"E");
outtextxy(18*x,8*y,"F");
outtextxy(31*x,31*y/4,"G");
outtextxy(41*x,8*y,"H");
outtextxy(28*x,15*y/2,"I");
outtextxy(41*x,37*y/4,"J");
outtextxy(50*x,9*y,"K");
outtextxy(38*x,11*y,"L");
outtextxy(50*x,23*y/2,"M");
/*******************************************************************/
moveto(10*x,5*y);   lineto(20*x,5*y);   lineto(20*x,8*y);
lineto(40*x,8*y);   lineto(40*x,11*y);   lineto(50*x,11*y);
moveto(10*x,5*y);   lineto(10*x,13*y/2); lineto(30*x,13*y/2);
lineto(30*x,19*y/2); lineto(50*x,19*y/2); lineto(50*x,11*y);
/*******************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
 while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,19*y,"    Please type a, b, c or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
   setcolor(backcolor);
   bar(50*x,37*y/2,88*x,20*y);
   setcolor(forecolor);
   }
  }
```

```
/*******************************************************************/
 switch (Ch)        {
 case 'a': outtextxy(47*x,19*y,"a");
   outtextxy(52*x,19*y,"You want to see the algorithm ");
   outtextxy(52*x,20*y,"again. Press any key to continue.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(50*x,37*y/2,179*x/2,21*y);
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   show_alg();
   break;
case 'b': outtextxy(47*x,19*y,"b");
   outtextxy(52*x,19*y,"You want to compare your solu-");
   outtextxy(52*x,20*y,"tion with ours. So press any  ");
   outtextxy(52*x,21*y,"key to see it.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(50*x,37*y/2,179*x/2,22*y);
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   compare_solutions();
   break;
case 'c': outtextxy(47*x,19*y,"c");
   outtextxy(52*x,19*y,"You want to see step by step");
   outtextxy(52*x,20*y,"solution. So press any key to ");
   outtextxy(52*x,21*y,"continue.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(50*x.37*y/2,179*x/2,22*y);
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   step_solution();
   break;
case 'd': outtextxy(47*x,19*y,"d");
   confirm_exit();
```

```
      break;
    default  : break;
   }
 }
 closegraph();
}




/***********************************************************************/
/* This routine gives breadth first search spanning tree algorithm     */
/***********************************************************************/
static void show_alg(void)
{
 outtextxy(15*x,12*y,"DEPTH  FIRST  SEARCH SPANNING TREE
                          ALGORITHM");
 outtextxy(2*x,13*y,"Step 1 . Pick a vertex x. L = { x } (list of vertices in the
                         tree)");
 outtextxy(2*x,14*y,"        T = 0 (list of edges in the tree), x <- 1, k <- 2
                         (counter)");
 outtextxy(2*x,15*y,"Step 2 . Pick any vertex U not in L, such that U is adjacent to
                         the");
 outtextxy(2*x,16*y,"        vertex L with the highest label, say V.");
 outtextxy(2*x,17*y,"        L = L U { U }, T = T U {U,V}, U <- k,  k <- k + 1");
 outtextxy(2*x,18*y,"Step 3 . a) If all vertices are in L, stop");
 outtextxy(2*x,19*y,"        b) If not all vertices are in L  ");
 outtextxy(2*x,20*y,"          1) If there exist a vertex adjacent not in L which is ");
 outtextxy(2*x,21*y,"            adjacent to a vertex in L, go to Step 2");
 outtextxy(2*x,22*y,"          2) If no such vertex exists, stop and output that the");
 outtextxy(2*x,23*y,"            graph is not connected.");
 Pause(30*x,24*y);
 setcolor(backcolor);
 bar(2*x,47*y/4,179*x/2,49*y/2);
 setcolor(forecolor);
}
```

```
/*********************************************************************/
/* This routine gives the solution to the exercise to be compared.       */
/*********************************************************************/
static void compare_solutions(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  /*******************************************************************/
  moveto(10*x,5*y);   lineto(20*x,5*y);   lineto(20*x,13*y/2);
  lineto(30*x,13*y/2); lineto(30*x,8*y);   lineto(40*x,8*y);
  lineto(40*x,19*y/2); lineto(50*x,19*y/2); lineto(50*x,11*y);
  lineto(40*x,11*y);
  moveto(10*x,13*y/2); lineto(20*x,13*y/2);
  moveto(20*x,8*y);   lineto(30*x,8*y);
  moveto(30*x,19*y/2); lineto(40*x,19*y/2);
  /*******************************************************************/
  setcolor(forecolor);
  setlinestyle(3,0,3);
  /*******************************************************************/
  moveto(10*x,5*y);   lineto(20*x,5*y);   lineto(20*x,13*y/2);
  lineto(30*x,13*y/2); lineto(30*x,8*y);   lineto(40*x,8*y);
  lineto(40*x,19*y/2); lineto(50*x,19*y/2); lineto(50*x,11*y);
  lineto(40*x,11*y);
  moveto(10*x,13*y/2); lineto(20*x,13*y/2);
  moveto(20*x,8*y);   lineto(30*x,8*y);
  moveto(30*x,19*y/2); lineto(40*x,19*y/2);
  outtextxy(6*x,5*y,"(1)");            /* A */
  outtextxy(21*x,5*y,"(2)");           /* B */
  outtextxy(45*x/2,25*y/4,"(3)");      /* D */
  outtextxy(33*x,13*y/2,"(4)");        /* E */
  outtextxy(33*x,31*y/4,"(5)");        /* G */
  outtextxy(42*x,8*y,"(6)");           /* H */
  outtextxy(36*x,9*y,"(7)");           /* J */
  outtextxy(46*x,9*y,"(8)");           /* K */
  outtextxy(46*x,23*y/2,"(9)");        /* M */
  outtextxy(39*x,23*y/2,"(10)");       /* L */
outtextxy(29*x,10*y,"(11)");         /* I */
outtextxy(19*x,17*y/2,"(12)");      /* F */
outtextxy(9*x,7*y,"(13)");          /* C */
/*********************************************************************/
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,17*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/*********************************************************************/
pieslice(10*x,5*y,0,359,2);        /* A */   /* redraw the graph */
pieslice(20*x,5*y,0,359,2);        /* B */
pieslice(10*x,13*y/2,0,359,2);   /* C */
pieslice(20*x,13*y/2,0,359,2);   /* D */
pieslice(30*x,13*y/2,0,359,2);   /* E */
pieslice(20*x,8*y,0,359,2);        /* F */
pieslice(30*x,8*y,0,359,2);        /* G */
pieslice(40*x,8*y,0,359,2);        /* H */
pieslice(30*x,19*y/2,0,359,2);   /* I */
pieslice(40*x,19*y/2,0,359,2);   /* J */
pieslice(50*x,19*y/2,0,359,2);   /* K */
pieslice(40*x,11*y,0,359,2);       /* L */
pieslice(50*x,11*y,0,359,2);      /* M */
/*********************************************************************/
outtextxy(10*x,9*y/2,"A");
outtextxy(20*x,9*y/2,"B");
outtextxy(8*x,13*y/2,"C");
outtextxy(21*x,25*y/4,"D");
outtextxy(31*x,13*y/2,"E");
outtextxy(18*x,8*y,"F");
outtextxy(31*x,31*y/4,"G");
outtextxy(41*x,8*y,"H");
outtextxy(28*x,19*y/2,"I");
outtextxy(41*x,37*y/4,"J");
outtextxy(50*x,9*y,"K");
outtextxy(38*x,11*y,"L");
```

```
   outtextxy(50*x,23*y/2,"M");
   /*******************************************************************/
   moveto(10*x,5*y);   lineto(20*x,5*y);   lineto(20*x,8*y);
   lineto(40*x,8*y);   lineto(40*x,11*y);  lineto(50*x,11*y);
   moveto(10*x,5*y);   lineto(10*x,13*y/2); lineto(30*x,13*y/2);
   lineto(30*x,19*y/2); lineto(50*x,19*y/2); lineto(50*x,11*y);
}

/*******************************************************************/
/* This routine gives the step by step solution to the exercise      */
/*******************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(3*x/2,5*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   /*******************************************************************/
   pieslice(10*x,5*y,0,359,2);       /* A */
   pieslice(20*x,5*y,0,359,2);       /* B */
   pieslice(10*x,13*y/2,0,359,2);    /* C */
   pieslice(20*x,13*y/2,0,359,2);    /* D */
   pieslice(30*x,13*y/2,0,359,2);    /* E */
   pieslice(20*x,8*y,0,359,2);       /* F */
   pieslice(30*x,8*y,0,359,2);       /* G */
   pieslice(40*x,8*y,0,359,2);       /* H */
   pieslice(30*x,19*y/2,0,359,2);    /* I */
   pieslice(40*x,19*y/2,0,359,2);    /* J */
   pieslice(50*x,19*y/2,0,359,2);    /* K */
   pieslice(40*x,11*y,0,359,2);      /* L */
   pieslice(50*x,11*y,0,359,2);      /* M */
   /*******************************************************************/
   outtextxy(10*x,9*y/2,"A");
   outtextxy(20*x,9*y/2,"B");
   outtextxy(8*x,13*y/2,"C");
   outtextxy(21*x,25*y/4,"D");
   outtextxy(31*x,13*y/2,"E");
outtextxy(18*x,8*y,"F");
outtextxy(31*x,31*y/4,"G");
outtextxy(41*x,8*y,"H");
outtextxy(28*x,19*y/2,"I");
outtextxy(41*x,37*y/4,"J");
outtextxy(50*x,9*y,"K");
outtextxy(38*x,11*y,"L");
outtextxy(50*x,23*y/2,"M");
/*****************************************************************/
moveto(10*x,5*y);  lineto(20*x,5*y);  lineto(20*x,8*y);
lineto(40*x,8*y);  lineto(40*x,11*y);  lineto(50*x,11*y);
moveto(10*x,5*y);  lineto(10*x,13*y/2); lineto(30*x,13*y/2);
lineto(30*x,19*y/2); lineto(50*x,19*y/2); lineto(50*x,11*y);
/*****************************************************************/
outtextxy(56*x,2*y,"k");
outtextxy(60*x,2*y,"V");
outtextxy(64*x,2*y,"U");
outtextxy(70*x,2*y,"L");
outtextxy(75*x,2*y,"Label");
outtextxy(86*x,2*y,"T");
moveto(55*x,5*y/2); lineto(58*x,5*y/2);
moveto(59*x,5*y/2); lineto(62*x,5*y/2);
moveto(63*x,5*y/2); lineto(66*x,5*y/2);
moveto(68*x,5*y/2); lineto(73*x,5*y/2);
moveto(74*x,5*y/2); lineto(165*x/2,5*y/2);
moveto(84*x,5*y/2); lineto(89*x,5*y/2);
/*****************************************************************/
outtextxy(70*x,3*y,"A");
outtextxy(75*x,3*y,"A <- 1");
outtextxy(6*x,5*y,"(1)");
outtextxy(56*x,4*y,"2");
/*****************************************************************/
Pause(30*x,24*y);
outtextxy(60*x,4*y,"A");
outtextxy(64*x,4*y,"B");
outtextxy(70*x,4*y,"B");
outtextxy(75*x,4*y,"B <- 2");
outtextxy(84*x,4*y,"(A,B)");
outtextxy(21*x,5*y,"(2)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(10*x,5*y); lineto(20*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(10*x,5*y); lineto(20*x,5*y);    /* add (A, B) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,5*y,"3");
/*********************************************************************/
outtextxy(60*x,5*y,"B");
outtextxy(64*x,5*y,"D");
outtextxy(70*x,5*y,"D");
outtextxy(75*x,5*y,"D <- 3");
outtextxy(84*x,5*y,"(B,D)");
outtextxy(45*x/2,25*y/4,"(3)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(20*x,5*y); lineto(20*x,13*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,5*y); lineto(20*x,13*y/2); /* add (B, D) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,6*y,"4");
/*********************************************************************/
outtextxy(60*x,6*y,"D");
outtextxy(64*x,6*y,"E");
outtextxy(70*x,6*y,"E");
outtextxy(75*x,6*y,"E <- 4");
outtextxy(84*x,6*y,"(D,E)");
outtextxy(33*x,13*y/2,"(4)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(20*x,13*y/2);  lineto(30*x,13*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,13*y/2);  lineto(30*x,13*y/2);  /* add (D, E) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,7*y,"5");
/*********************************************************************/
outtextxy(60*x,7*y,"E");
outtextxy(64*x,7*y,"G");
outtextxy(70*x,7*y,"G");
outtextxy(75*x,7*y,"G <- 5");
outtextxy(84*x,7*y,"(E,G)");
outtextxy(33*x,31*y/4,"(5)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(30*x,13*y/2);  lineto(30*x,8*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,13*y/2);  lineto(30*x,8*y);     /* add (E, G) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,8*y,"6");
```

```
/*********************************************************************/
outtextxy(60*x,8*y,"G");
outtextxy(64*x,8*y,"H");
outtextxy(70*x,8*y,"H");
outtextxy(75*x,8*y,"H <- 6");
outtextxy(42*x,8*y,"(6)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(30*x,8*y); lineto(40*x,8*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,8*y); lineto(40*x,8*y);    /* add (G, H) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,9*y,"7");
/*********************************************************************/
outtextxy(60*x,9*y,"H");
outtextxy(64*x,9*y,"J");
outtextxy(70*x,9*y,"J");
outtextxy(75*x,9*y,"J <- 7");
outtextxy(84*x,9*y,"(H,J)");
outtextxy(36*x,9*y,"(7)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(40*x,8*y); lineto(40*x,19*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,8*y); lineto(40*x,19*y/2);    /* add (H, J) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
```

```
outtextxy(56*x,10*y,"8");
/*****************************************************************/
outtextxy(60*x,10*y,"J");
outtextxy(64*x,10*y,"K");
outtextxy(70*x,10*y,"K");
outtextxy(75*x,10*y,"K <- 8");
outtextxy(84*x,10*y,"(J,K)");
outtextxy(46*x,9*y,"(8)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(40*x,19*y/2);  lineto(50*x,19*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,19*y/2);  lineto(50*x,19*y/2);     /* add (J, K) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,11*y,"9");
/*****************************************************************/
outtextxy(60*x,11*y,"K");
outtextxy(64*x,11*y,"M");
outtextxy(70*x,11*y,"M");
outtextxy(75*x,11*y,"M <- 9");
outtextxy(84*x,11*y,"(K,M)");
outtextxy(46*x,23*y/2,"(9)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(50*x,19*y/2);  lineto(50*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,19*y/2);  lineto(50*x,11*y);       /* add (K, M) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
```

```
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,12*y,"10");
/********************************************************************/
outtextxy(60*x,12*y,"M");
outtextxy(64*x,12*y,"L");
outtextxy(70*x,12*y,"L");
outtextxy(75*x,12*y,"L <- 10");
outtextxy(84*x,12*y,"(M,L)");
outtextxy(39*x,23*y/2,"(10)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(40*x,11*y);  lineto(50*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,11*y);  lineto(50*x,11*y);      /* add (M, L) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,13*y,"11");
/********************************************************************/
outtextxy(60*x,13*y,"L");
outtextxy(64*x,13*y,"-");
outtextxy(60*x,14*y,"M");
outtextxy(64*x,14*y,"-");
outtextxy(60*x,15*y,"K");
outtextxy(64*x,15*y,"-");
outtextxy(60*x,16*y,"J");
outtextxy(64*x,16*y,"I");
outtextxy(70*x,16*y,"I");
outtextxy(75*x,16*y,"I <- 11");
outtextxy(84*x,16*y,"(J,I)");
outtextxy(29*x,10*y,"(11)");
Pause(30*x,24*y);
```

```
setcolor(backcolor);
moveto(30*x,19*y/2); lineto(40*x,19*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,19*y/2); lineto(40*x,19*y/2);      /* add (J, I) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,17*y,"12");
/*****************************************************************/
outtextxy(60*x,17*y,"I");
outtextxy(64*x,17*y,"-");
outtextxy(60*x,18*y,"J");
outtextxy(64*x,18*y,"-");
outtextxy(60*x,19*y,"H");
outtextxy(64*x,19*y,"-");
outtextxy(60*x,20*y,"G");
outtextxy(64*x,20*y,"F");
outtextxy(70*x,20*y,"F");
outtextxy(75*x,20*y,"F <- 12");
outtextxy(84*x,20*y,"(G,F)");
outtextxy(19*x,17*y/2,"(12)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(20*x,8*y); lineto(30*x,8*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,8*y); lineto(30*x,8*y);      /* add (G, F) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,21*y,"13");
```

```
/*****************************************************************/
outtextxy(60*x,21*y,"F");
outtextxy(64*x,21*y,"-");
outtextxy(60*x,22*y,"G");
outtextxy(64*x,22*y,"-");
outtextxy(60*x,23*y,"E");
outtextxy(64*x,23*y,"-");
outtextxy(60*x,24*y,"D");
outtextxy(64*x,24*y,"C");
outtextxy(70*x,24*y,"C");
outtextxy(75*x,24*y,"C <- 13");
outtextxy(84*x,24*y,"(D,C)");
outtextxy(9*x,7*y,"(13)");
Pause(15*x,24*y);
setcolor(backcolor);
moveto(10*x,13*y/2);  lineto(20*x,13*y/2);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(10*x,13*y/2);  lineto(20*x,13*y/2);      /* add (D, C) to T */
setlinestyle(0,0,3);
Pause(15*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,55*x,49*y/2);
setcolor(forecolor);
outtextxy(35*x,23*y,"We are done.");
/*****************************************************************/
Pause(15*x,24*y);
setcolor(backcolor);          /* Clean the game field  again */
bar(3*x/2,5*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,2*y,"Use the depth first search algorithm to find a minimal spanning
                   tree.");
outtextxy(2*x,3*y,"(Start at A. If there is a choice of edges select edges according
                   to");
outtextxy(2*x,4*y,"alphabetical order.)");
```

```
/*********************************************************************/
pieslice(10*x,5*y,0,359,2);      /* A */  /* redraw the graph */
pieslice(20*x,5*y,0,359,2);      /* B */
pieslice(10*x,13*y/2,0,359,2);   /* C */
pieslice(20*x,13*y/2,0,359,2);   /* D */
pieslice(30*x,13*y/2,0,359,2);   /* E */
pieslice(20*x,8*y,0,359,2);      /* F */
pieslice(30*x,8*y,0,359,2);      /* G */
pieslice(40*x,8*y,0,359,2);      /* H */
pieslice(30*x,19*y/2,0,359,2);   /* I */
pieslice(40*x,19*y/2,0,359,2);   /* J */
pieslice(50*x,19*y/2,0,359,2);   /* K */
pieslice(40*x,11*y,0,359,2);     /* L */
pieslice(50*x,11*y,0,359,2);     /* M */
/*********************************************************************/
outtextxy(10*x,9*y/2,"A");
outtextxy(20*x,9*y/2,"B");
outtextxy(8*x,13*y/2,"C");
outtextxy(21*x,25*y/4,"D");
outtextxy(31*x,13*y/2,"E");
outtextxy(18*x,8*y,"F");
outtextxy(31*x,31*y/4,"G");
outtextxy(41*x,8*y,"H");
outtextxy(28*x,19*y/2,"I");
outtextxy(41*x,37*y/4,"J");
outtextxy(50*x,9*y,"K");
outtextxy(38*x,11*y,"L");
outtextxy(50*x,23*y/2,"M");
/*********************************************************************/
moveto(10*x,5*y);  lineto(20*x,5*y);  lineto(20*x,8*y);
lineto(40*x,8*y);  lineto(40*x,11*y);  lineto(50*x,11*y);
moveto(10*x,5*y);  lineto(10*x,13*y/2); lineto(30*x,13*y/2);
lineto(30*x,19*y/2); lineto(50*x,19*y/2); lineto(50*x,11*y);
}
```

```
/**********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
  }
  switch (ch)        {
   case 'y': in_the_exercise = 0;
         break;
   case 'Y': in_the_exercise = 0;
         break;
   case 'n': setcolor(backcolor);
         bar(46*x,37*y/2,179*x/2,22*y);
         setcolor(forecolor);
         break;
   case 'N': setcolor(backcolor);
         bar(46*x,37*y/2,179*x/2,22*y);
         setcolor(forecolor);
         break;
  default : break;
  }
}
```

```
/* PROGRAM  : q432.c
  AUTHOR     : Atilla BAKAN
  DATE       : Apr. 7, 1990
  REVISED    : Apr. 17, 1990

  DESCRIPTION : This program contains the second exercise about the
                depth first search for spanning trees.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)     _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk          find_t
  #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name          name
  #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions        */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer           (void);
static void example        (void);
static void show_alg       (void);
static void step_solution   (void);
static void compare_solutions (void);
static void confirm_exit    (void);

/************************************************************************/
/* miscellaneous global variables                                       */
/************************************************************************/
 int in_the_exercise = 1;


/************************************************************************/
/* graphic initialization variables                                     */
/************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code          */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines                    */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**********************************************************************/
  settext();
  /**********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }



/*********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
   switch (ch)         {
    case 'y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}


/************************************************************************/
/* This function sets the text default values                          */
/************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/***********************************************************************/
/* main routine which calls exer routine                               */
/***********************************************************************/
void main()
{
   exer();
}
```

```
/**********************************************************************/
/* This routine  asks the question, then depending on the user's answer          */
/* makes necessary explanations                                                  */
/**********************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  2");
  /*******************************************************************/
  outtextxy(2*x,2*y,"Use the depth first search algorithm to find a minimal spanning
                  tree.");
  outtextxy(2*x,3*y,"(Start at A. If there is a choice of edges select edges according
                  to");
  outtextxy(2*x,4*y,"alphabetical order.)");
  /*******************************************************************/
  pieslice(25*x,5*y,0,359,2);     /* A */
  pieslice(25*x,11*y,0,359,2);    /* B */
  pieslice(55*x,5*y,0,359,2);     /* C */
  pieslice(55*x,11*y,0,359,2);    /* D */
  pieslice(35*x,7*y,0,359,2);     /* E */
  pieslice(45*x,7*y,0,359,2);     /* F */
  pieslice(35*x,9*y,0,359,2);     /* G */
  pieslice(45*x,9*y,0,359,2);     /* H */
  /*******************************************************************/
  outtextxy(25*x,9*y/2,"A");
  outtextxy(25*x,23*y/2,"B");
  outtextxy(55*x,9*y/2,"C");
  outtextxy(55*x,23*y/2,"D");
  outtextxy(33*x,7*y,"E");
  outtextxy(46*x,7*y,"F");
  outtextxy(33*x,9*y,"G");
outtextxy(46*x,9*y,"H");
/*********************************************************************/
moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
lineto(45*x,7*y);  lineto(35*x,7*y); lineto(35*x,9*y);
lineto(45*x,9*y);
moveto(45*x,7*y);lineto(35*x,9*y);
moveto(25*x,5*y);lineto(35*x,7*y);
/*********************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
    switch (Ch)         {
    case 'a': outtextxy(47*x,19*y,"a");
     outtextxy(52*x,19*y,"You want to see the algorithm ");
     outtextxy(52*x,20*y,"again. Press any key to continue.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,37*y/2,179*x/2,21*y);
     bar(2*x,13*y,179*x/2,49*y/2);
```

```
        setcolor(forecolor);
        show_alg();
        break;
      case 'b': outtextxy(47*x,19*y,"b");
        outtextxy(52*x,19*y,"You want to compare your solu-");
        outtextxy(52*x,20*y,"tion with ours. So press any  ");
        outtextxy(52*x,21*y,"key to see it.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        compare_solutions();
        break;
      case 'c': outtextxy(47*x,19*y,"c");
        outtextxy(52*x,19*y,"You want to see step by step");
        outtextxy(52*x,20*y,"solution. So press any key to ");
        outtextxy(52*x,21*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
    }
  }
  closegraph();
}
```

```
/*************************************************************************/
/* This routine gives breadth first search spanning tree algorithm           */
/*************************************************************************/
static void show_alg(void)
{
  outtextxy(15*x,12*y,"DEPTH  FIRST  SEARCH SPANNING TREE
                         ALGORITHM");
  outtextxy(2*x,13*y,"Step 1 . Pick a vertex x. L = { x } (list of vertices in the
                         tree)");
  outtextxy(2*x,14*y,"       T = 0 (list of edges in the tree), x <- 1, k <- 2
                         (counter)");
  outtextxy(2*x,15*y,"Step 2 . Pick any vertex U not in L, such that U is adjacent to
                         the");
  outtextxy(2*x,16*y,"       vertex L with the highest label, say V.");
  outtextxy(2*x,17*y,"       L = L U { U }, T = T U {U,V}, U <- k,  k <- k + 1");
  outtextxy(2*x,18*y,"Step 3 . a) If all vertices are in L, stop");
  outtextxy(2*x,19*y,"       b) If not all vertices are in L  ");
  outtextxy(2*x,20*y,"         1) If there exist a vertex adjacent not in L which is ");
  outtextxy(2*x,21*y,"           adjacent to a vertex in L, go to Step 2");
  outtextxy(2*x,22*y,"         2) If no such vertex exists, stop and output that the");
  outtextxy(2*x,23*y,"           graph is not connected.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/*******************************************************************/
/* This routine gives the solution to the exercise to be compared.          */
/*******************************************************************/
static void compare_solutions(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  /*****************************************************************/
  moveto(25*x,5*y);  lineto(25*x,11*y);
  lineto(55*x,11*y); lineto(55*x,5*y);
  moveto(55*x,11*y); lineto(45*x,9*y);
  lineto(45*x,7*y);  lineto(35*x,7*y);
  lineto(35*x,9*y);
  /*****************************************************************/
  setcolor(forecolor);
  /*****************************************************************/
  outtextxy(27*x,9*y/2,"(1)");
  outtextxy(22*x,11*y,"(2)");
  outtextxy(56*x,11*y,"(3)");
  outtextxy(56*x,5*y,"(4)");
  outtextxy(47*x,9*y,"(5)");
  outtextxy(43*x,13*y/2,"(6)");
  outtextxy(36*x,13*y/2,"(7)");
  outtextxy(36*x,19*y/2,"(8)");
  /*****************************************************************/
  setlinestyle(3,0,3);
  /*****************************************************************/
  moveto(25*x,5*y);  lineto(25*x,11*y);
  lineto(55*x,11*y); lineto(55*x,5*y);
  moveto(55*x,11*y); lineto(45*x,9*y);
  lineto(45*x,7*y);  lineto(35*x,7*y);
  lineto(35*x,9*y);
  /*****************************************************************/
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
```

```
    bar(3*x/2,17*y/4,179*x/2,49*y/2);
    setcolor(forecolor);
    /***********************************************************************/
    pieslice(25*x,5*y,0,359,2);      /* A */
    pieslice(25*x,11*y,0,359,2);     /* B */
    pieslice(55*x,5*y,0,359,2);      /* C */
    pieslice(55*x,11*y,0,359,2);     /* D */
    pieslice(35*x,7*y,0,359,2);      /* E */
    pieslice(45*x,7*y,0,359,2);      /* F */
    pieslice(35*x,9*y,0,359,2);      /* G */
    pieslice(45*x,9*y,0,359,2);      /* H */
    /***********************************************************************/
    outtextxy(25*x,9*y/2,"A");
    outtextxy(25*x,23*y/2,"B");
    outtextxy(55*x,9*y/2,"C");
    outtextxy(55*x,23*y/2,"D");
    outtextxy(33*x,7*y,"E");
    outtextxy(46*x,7*y,"F");
    outtextxy(33*x,9*y,"G");
    outtextxy(46*x,9*y,"H");
    /***********************************************************************/
    moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
    lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
    lineto(45*x,7*y);  lineto(35*x,7*y); lineto(35*x,9*y);
    lineto(45*x,9*y);
    moveto(45*x,7*y);lineto(35*x,9*y);
    moveto(25*x,5*y);lineto(35*x,7*y);
}
```

```
/************************************************************************/
/* This routine gives the step by step solution to the exercise          */
/************************************************************************/
static void step_solu*ion(void)
{

  setcolor(backcolor);        /* Clean the game field */
  bar(3*x/2,17*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
 /**********************************************************************/
  pieslice(20*x,5*y,0,359,2);     /* A */
  pieslice(20*x,11*y,0,359,2);    /* B */
  pieslice(50*x,5*y,0,359,2);     /* C */
  pieslice(50*x,11*y,0,359,2);    /* D */
  pieslice(30*x,7*y,0,359,2);     /* E */
  pieslice(40*x,7*y,0,359,2);     /* F */
  pieslice(30*x,9*y,0,359,2);     /* G */
  pieslice(40*x,9*y,0,359,2);     /* H */
 /**********************************************************************/
  outtextxy(20*x,9*y/2,"A");
  outtextxy(20*x,23*y/2,"B");
  outtextxy(50*x,9*y/2,"C");
  outtextxy(50*x,23*y/2,"D");
  outtextxy(28*x,7*y,"E");
  outtextxy(42*x,7*y,"F");
  outtextxy(28*x,9*y,"G");
  outtextxy(41*x,9*y,"H");
 /**********************************************************************/
  moveto(50*x,11*y); lineto(50*x,5*y); lineto(20*x,5*y);
  lineto(20*x,11*y); lineto(50*x,11*y);lineto(40*x,9*y);
  lineto(40*x,7*y);  lineto(30*x,7*y); lineto(30*x,9*y);
  lineto(40*x,9*y);
  moveto(40*x,7*y);lineto(30*x,9*y);
  moveto(20*x,5*y);lineto(30*x,7*y);
  /*********************************************************************/
  outtextxy(56*x,5*y,"k");
outtextxy(60*x,5*y,"V");
outtextxy(64*x,5*y,"U");
outtextxy(70*x,5*y,"L");
outtextxy(75*x,5*y,"Label");
outtextxy(86*x,5*y,"T");
moveto(55*x,11*y/2); lineto(58*x,11*y/2);
moveto(59*x,11*y/2); lineto(62*x,11*y/2);
moveto(63*x,11*y/2); lineto(66*x,11*y/2);
moveto(68*x,11*y/2); lineto(73*x,11*y/2);
moveto(74*x,11*y/2); lineto(165*x/2,11*y/2);
moveto(84*x,11*y/2); lineto(89*x,11*y/2);
/*******************************************************************/
outtextxy(70*x,6*y,"A");
outtextxy(75*x,6*y,"A <- 1");
outtextxy(22*x,9*y/2,"(1)");
outtextxy(56*x,7*y,"2");
/*******************************************************************/
Pause(30*x,24*y);
outtextxy(60*x,7*y,"A");
outtextxy(64*x,7*y,"B");
outtextxy(70*x,7*y,"B");
outtextxy(75*x,7*y,"B <- 2");
outtextxy(84*x,7*y,"(A,B)");
outtextxy(16*x,11*y,"(2)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(20*x,5*y); lineto(20*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,5*y); lineto(20*x,11*y);    /* add (A, B) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,8*y,"3");
```

```
/*****************************************************************/
outtextxy(60*x,8*y,"B");
outtextxy(64*x,8*y,"D");
outtextxy(70*x,8*y,"D");
outtextxy(75*x,8*y,"D <- 3");
outtextxy(84*x,8*y,"(B,D)");
outtextxy(51*x,11*y,"(3)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(20*x,11*y);  lineto(50*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,11*y);  lineto(50*x,11*y);  /* add (B, D) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,9*y,"4");
/*****************************************************************/
outtextxy(60*x,9*y,"D");
outtextxy(64*x,9*y,"C");
outtextxy(70*x,9*y,"C");
outtextxy(75*x,9*y,"C <- 4");
outtextxy(84*x,9*y,"(D,C)");
outtextxy(51*x,5*y,"(4)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(50*x,11*y);  lineto(50*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,11*y);  lineto(50*x,5*y);  /* add (D, C) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
```

```
setcolor(forecolor);
outtextxy(56*x,10*y,"5");
/*****************************************************************/
outtextxy(60*x,10*y,"C");
outtextxy(64*x,10*y,"-");
outtextxy(60*x,11*y,"D");
outtextxy(64*x,11*y,"H");
outtextxy(70*x,11*y,"H");
outtextxy(75*x,11*y,"H <- 5");
outtextxy(84*x,11*y,"(D,H)");
outtextxy(42*x,9*y,"(5)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(50*x,11*y); lineto(40*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,11*y); lineto(40*x,9*y);    /* add (D, H) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,12*y,"6");
/*****************************************************************/
outtextxy(60*x,12*y,"H");
outtextxy(64*x,12*y,"F");
outtextxy(70*x,12*y,"F");
outtextxy(75*x,12*y,"F <- 6");
outtextxy(38*x,13*y/2,"(6)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(40*x,9*y); lineto(40*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,9*y); lineto(40*x,7*y);    /* add (H, F) to T */
setlinestyle(0,0,3);
```

```
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,13*y,"7");
/*******************************************************************/
outtextxy(60*x,13*y,"F");
outtextxy(64*x,13*y,"E");
outtextxy(70*x,13*y,"E");
outtextxy(75*x,13*y,"E <- 7");
outtextxy(84*x,13*y,"(F,E)");
outtextxy(29*x,13*y/2,"(7)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(40*x,7*y);  lineto(30*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,7*y);  lineto(30*x,7*y);    /* add (F, E) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,14*y,"8");
/*******************************************************************/
outtextxy(60*x,14*y,"E");
outtextxy(64*x,14*y,"G");
outtextxy(70*x,14*y,"G");
outtextxy(75*x,14*y,"G <- 8");
outtextxy(84*x,14*y,"(E,G)");
outtextxy(31*x,19*y/2,"(8)");
Pause(30*x,24*y);
setcolor(backcolor);
moveto(30*x,7*y);  lineto(30*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
```

```
moveto(30*x,7*y); lineto(30*x,9*y);    /* add (E, G) to T */
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,80*x,49*y/2);
setcolor(forecolor);
outtextxy(56*x,11*y,"9");
/*******************************************************************/
outtextxy(60*x,16*y,"We are done.");
/*******************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);          /* Clean the game field  again */
bar(3*x/2,17*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(25*x,5*y,0,359,2);     /* A */
pieslice(25*x,11*y,0,359,2);    /* B */
pieslice(55*x,5*y,0,359,2);     /* C */
pieslice(55*x,11*y,0,359,2);    /* D */
pieslice(35*x,7*y,0,359,2);     /* E */
pieslice(45*x,7*y,0,359,2);     /* F */
pieslice(35*x,9*y,0,359,2);     /* G */
pieslice(45*x,9*y,0,359,2);     /* H */
/*******************************************************************/
outtextxy(25*x,9*y/2,"A");
outtextxy(25*x,23*y/2,"B");
outtextxy(55*x,9*y/2,"C");
outtextxy(55*x,23*y/2,"D");
outtextxy(33*x,7*y,"E");
outtextxy(46*x,7*y,"F");
outtextxy(33*x,9*y,"G");
outtextxy(46*x,9*y,"H");
/*******************************************************************/
moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
lineto(45*x,7*y);  lineto(35*x,7*y); lineto(35*x,9*y);
```

```
    lineto(45*x,9*y);
    moveto(45*x,7*y);lineto(35*x,9*y);
    moveto(25*x,5*y);lineto(35*x,7*y);
}
```

```
/***********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
    switch (ch)         {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;
    case 'n': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;
    case 'N': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;
    default : break;
    }
}
```

```
/* PROGRAM  : minimal.c
   AUTHOR   : Atilla BAKAN
   DATE     : Feb. 14, 1990
   REVISED  : Apr. 18, 1990

   DESCRIPTION : This program contains the tutorial for minimal spanning
                 trees. It has two algoritms, namely Prim's Algorithm and
                 Kruskal's algorithm. For each algorithm two examples are
                 solved step by step.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
```

```
    #define ffblk          FIND
    #define ff_name          name
    #define ff_attrib       attribute
#endif

#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void add_shadow    (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit    (int errnum);
static void initialize    (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void Pageup        (void);
static void Pagedown      (void);
static void press_a_key   (int wrow);
static void pre_help      (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);

/* Tutorial procedures            */
static void minimal_spanning_trees(void);
static void definition_4_4_1(void);
static void example_4_4_1 (void);
static void prim_alg      (void);
static void alg_ex_prim_1 (void);
static void alg_ex_prim_2 (void);
static void kruskals_alg  (void);
static void ex_kruskal_1 (void);
static void ex_kruskal_2 (void);
static void exercises    (void);
static void exer1        (void);
static void exer2        (void);
static void exer3        (void);
static void exer4        (void);
static void P1          (void);
static void P2          (void);
static void P3          (void);
static void P4          (void);
static void P5          (void);
static void P6          (void);
static void P7          (void);
static void P8          (void);
static void P9          (void);
static void P10          (void);
static void P11          (void);
static void P12          (void);
static void P13          (void);
static void P14          (void);

/*************************************************************************/
/* miscellaneous global variables                                       */
/*************************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];

/*************************************************************************/
/* error message table                                                  */
/*************************************************************************/
static char *error_text[]= {
   NULL,  /* errnum =  0, no error    */
   NULL,  /* errnum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
```

```
    "\t -m = force monochrome text attributes",
  "Memory allocation error"
};

/*********************************************************************/
/* miscellaneous defines                                             */
/*********************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33

/*********************************************************************/
/* this function will add a shadow to the active window              */
/*********************************************************************/
static void add_shadow(void)
{
  wshadow(LGREY|_BLACK);
}

/*********************************************************************/
/* this function pops open a window and confirms that the user really */
/* wants to quit the demo.  If so, it terminates the demo program.    */
/*********************************************************************/
static void confirm_quit(void)
{
  struct _onkey_t *kblist;

  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
  add_shadow():
  wputs("\n Quit demo, are you sure? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') normal_exit();
  wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
}

/*************************************************************************/
/* this function is called by the pull-down demo for a prompt            */
/*************************************************************************/
static void disp_sure_msg(void)
{
  wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
/*************************************************************************/
/* this function handles abnormal termination. If it is passed an        */
/* error code of 1, then it is a windowing system error. Otherwise       */
/* the error message is looked up in the error message table.            */
/*************************************************************************/
static void error_exit(int errnum)
{
  if(errnum) {
    printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
          exit(errnum);
  }
}
/*************************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems */
/*************************************************************************/
static void initialize(void)
{
  /* initialize the CXL video system and save current screen info */
  videoinit();
  readcur(&crow,&ccol);
  if((savescrn=ssave())==NULL) error_exit(3);

  /* if mouse exists, turn on full mouse support */
  if(msinit()) {
    mssupport(MS_FULL);
```

```
        msgotoxy(12,49);
    }

    /* attach [Alt-X] to the confirm_quit() function */
    setonkey(0x2d00,confirm_quit,0);

    /* attach [Ctrl Pageup] to the Pageup() function */
    setonkey(0x8400,Pageup,0);

    /* attach [Ctrl Pagedown] to the Pagedown() function */
    setonkey(0x7600,Pagedown,0);

    /* initialize help system, help key = [F1] */
  whelpdef("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
  WHITE|_RED.RED|_LGREY,pre_help);
}
/*************************************************************************/
/* this function is called anytime to switch back to previous window.    */
/*************************************************************************/
static void Pageup(void)
{
    static WINDOW handle;

    handle = whandle();
    wactiv(handle - 1);
}
/*************************************************************************/
/* this function is called anytime to switch back to next window.        */
/*************************************************************************/
static void Pagedown(void)
{
    static WINDOW handle;

    handle = whandle();
    wactiv(handle + 1);
}
```

```
/************************************************************************/
static void pre_help(void)
{
  add_shadow();
  setonkey(0x2d00,confirm_quit,0);
}
/************************************************************************/
/* this function handles normal termination. The original screen and cursor      */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.               */
/************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}

/************************************************************************/
/* this function displays a pause message then pauses for a keypress             */
/************************************************************************/
static void press_a_key(int wrow)
{
  register int attr1;
  register int attr2;

  attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
  attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
  wcenters(wrow,attr1,"Press a key");
  wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
  hidecur();
  if(waitkey()==ESC) confirm_quit();
  wcenters(wrow,attr1,"           ");
  wprints(wrow,0,attr2,"         ");
}
```

```
/**********************************************************************/
/* This routine causes short dealys during execution                  */
/**********************************************************************/
static void short_delay(void)
{
  delay_(SHORT_DELAY);
}

/**********************************************************************/
/* this function is called by the pull-down menu demo anytime         */
/* the  selection bar moves on or off the [Q]uit menu items.          */
/**********************************************************************/
static void quit_window(void)
{
  static WINDOW handle=0;

  if(handle) {
    wactiv(handle);
    wclose();
    handle=0;
  }
  else {
    handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
    wputs(" Quit takes you back to the\n demo program's main menu.");
  }
}

/**********************************************************************/
/* shows the cursor again if it has been hidden                       */
/**********************************************************************/
static void restore_cursor(void)
{
  wtextattr(WHITE|_MAGENTA);
  showcur();
}
```

```
/**********************************************************************/
/* enlarges or shrinks the windows                                    */
/**********************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
/**********************************************************************/
/* moves the active window to a given screen coordinates             */
/**********************************************************************/
static void move_window(int nsrow,int nscol)
{
   if(wmove(nsrow,nscol)) error_exit(1);
   short_delay();
}

/**********************************************************************/
/* this routine that calls minimal spanning trees() routine whenever Pageup   */
/* or pagdown keys are pressed.                                       */
/**********************************************************************/
void P1()
{
   wcloseall();
   minimal_spanning_trees();
}

/**********************************************************************/
/* this routine that calls example 4-4-1 routine whenever Pageup or   */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P2()
{
   wcloseall();
   example_4_4_1();
}
```

```
/*************************************************************************/
/* this routine that calls definition 4-4-1 routine whenever Pageup or     */
/* Pagedown keys are pressed.                                              */
/*************************************************************************/
void P3()
{
  wcloseall();
  definition_4_4_1();
}
/*************************************************************************/
/* this routine that calls prim_alg()  routine whenever Pageup or          */
/* Pagedown keys are pressed.                                              */
/*************************************************************************/
void P4()
{
  wcloseall();
  prim_alg();
}
/*************************************************************************/
/* this routine that calls alg_ex_prim_1 routine whenever Pageup or        */
/* Pagedown keys are pressed.                                              */
/*************************************************************************/
void P5()
{
  wcloseall();
  alg_ex_prim_1();
}
/*************************************************************************/
/* this routine that calls alg_ex_prim_2 routine whenever Pageup or        */
/* Pagedown keys are pressed.                                              */
/*************************************************************************/
void P6()
{
  wcloseall();
  alg_ex_prim_2();
}
```

```
/***********************************************************************/
/* this routine that calls kruskals_alg routine whenever Pageup or         */
/* Pagedown keys are pressed.                                               */
/***********************************************************************/
void P7()
{
  wcloseall();
  kruskals_alg();
}
/***********************************************************************/
/* this routine that calls ex_kruskal_1  routine whenever Pageup or        */
/* Pagedown keys are pressed.                                               */
/***********************************************************************/
void P8()
{
  wcloseall();
  ex_kruskal_1();
}
/***********************************************************************/
/* this routine that calls ex_kruskal_2 routine whenever Pageup or         */
/* Pagedown keys are pressed.                                               */
/***********************************************************************/
void P9()
{
  wcloseall();
  ex_kruskal_2();
}
/***********************************************************************/
/* this routine that calls exercises routine whenever Pageup or            */
/* Pagedown keys are pressed.                                               */
/***********************************************************************/
void P10()
{
  wcloseall();
  exercises();
}
```

```
/**********************************************************************/
/* this routine that calls exer1 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P11()
{
  wcloseall();
  exer1();
}
/**********************************************************************/
/* this routine that calls exer2 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P12()
{
  wcloseall();
  exer2();
}
/**********************************************************************/
/* this routine that calls exer3 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P13()
{
  wcloseall();
  exer3();
}
/**********************************************************************/
/* this routine that calls exer4 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P14()
{
  wcloseall();
  exer4();
}
```

```
/************************************************************************/
/* main routine that calls minimal spanning tree tutorial               */
/************************************************************************/
void main()
{
  initialize();
  minimal_spanning_trees();
}


/************************************************************************/
/* Routine that calls definition, example and algorithm routines about  */
/* minimal spanning trees.                                              */
/************************************************************************/
static void minimal_spanning_trees(void)
{
   register int *scrn;

   if((scrn=ssave())==NULL) error_exit(3);
   cclrscrn(LGREY|_BLUE);
   /*********************************************************************/
   /* attach [Pagedown] to the example_4_4_1() function */
   setonkey(0x5100,P2,0);
   /*********************************************************************/
   if((w[1]=wopen(6,15,11,54,3,LCYAN|_GREEN,BLACK|_GREEN))==0)
            error_exit(1);
   wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputsw(" We will try to introduce spanning tree concept with an example");
   press_a_key(3);
   wslide(0,0);
   short_delay();
   example_4_4_1();
   srestore(scrn);
}
```

```
/*************************************************************************/
/* This routine gives an example step by step implementation of Prim'          */
/* algorithm.                                                                    */
/*************************************************************************/
static void example_4_4_1 (void)
{
  /**********************************************************************/
  /* attach [Pageup] to the minimal_spanning_trees() function */
  setonkey(0x4900,P1,0);
  /**********************************************************************/
  /* attach [Pagedown] to the definition_4_4_1() function */
  setonkey(0x5100,P3,0);
  /**********************************************************************/
  if((w[2]=wopen(6,15,11,54,3,RED|_LGREY,BLACK|_MAGENTA))==0)
          error_exit(1);
  wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|YELLOW);
  add_shadow();
  wputsw(" Now consider a map. There are towns and roads between these towns"
          " Can you think of the situation as a graph?");
  press_a_key(3);
  wslide(0,39);
  short_delay();
  /**********************************************************************/
  if((w[3]=wopen(6,15,10,65,3,BLACK|_CYAN,RED|_LGREY))==0) error_exit(1);
  wtitle("[Minimal Spanning Trees - Example_4_4_1]",
         TCENTER,_LGREY|LBLUE);
  add_shadow();
  wputs("\n              To see the graph ");
  press_a_key(2);
  wcloseall();
  spawnl(P_WAIT,"examp441.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  definition_4_4_1();
}
```

```
/*************************************************************************/
/* Routine that gives the definition of minimal spanning trees.          */
/*************************************************************************/
static void definition_4_4_1(void)
{
  /**********************************************************************/
  /* attach [Pageup] to the example_4_4_1() function */
  setonkey(0x4900,P2,0);
  /**********************************************************************/
  /* attach [Pagedown] to the prim_alg() function */
  setonkey(0x5100,P4,0);
  /**********************************************************************/
  if((w[4]=wopen(6,20,16,58,3,BLACK|_CYAN,RED|_LGREY))==0) error_exit(1);
  wtitle("[Minimal Spanning Trees - Definition_4_4_1]",
         TCENTER,_LGREY|LBLUE);
  add_shadow();
  wputsw("  A minimal spanning tree in a weighted graph is a spanning"
           " tree for which the weight of the tree is as small as possible.");
  wputs("\n\n");
  wputsw(" In other words, a minimal spanning tree is a spanning tree such"
           " that no other spanning tree has a smaller weight.");
  press_a_key(8);
  wclose();
  /**********************************************************************/
  if((w[4]=wopen(6,20,12,58,3,GREEN|_BLACK,BLACK|_RED))==0)
           error_exit(1);
  wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|LBLUE);
  add_shadow();
  wputsw(" Now we will introduce you two algorithms to solve this type"
           " of problems. First one is called Prim's Algorithm.");
  press_a_key(4);
  wslide(0,0);
  prim_alg();
}
```

```
/**********************************************************************/
/* routine that introduces the Prim's minimal spanning tree algorithm          */
/**********************************************************************/
static void prim_alg(void)
{
  /**********************************************************************/
  /* attach [Pageup] to the definition_4_4_1() function */
  setonkey(0x4900,P3,0);
  /**********************************************************************/
  /* attach [Pagedown] to the alg_ex_prim_1() function */
  setonkey(0x5100,P5,0);
  /**********************************************************************/
  if((w[5]=wopen(0,15,24,65,3,BLACK|_GREEN,RED|_BLACK))==0)
            error_exit(1);
  wtitle("[Prim's Minimal Spanning Trees Algorithm]",TCENTER,BLUE|_LGREY);
  add_shadow();
  wputsw("The Method in this algorithm as briefly, is as follows :");
  wputs("\n\n");
  wputsw("  . Creates a set L of the vertices of the tree T  in the"
      " order it examined them");
  wputs("\n");
  wputsw("  . Build the tree by examining all edges from all vertices"
      " which is already in L to the ones which are not included "
      " L yet, and chooses the one with min weight to add too the"
      " minimal spanning tree. ");
  wputs("\n\n");
  wputsw(" Now, the actual algorithm is as follows :");
  wputs("\n");
  wputsw(" Step 1 . Pick an arbitrary initial vertex x.");
  wputs("\n          L = { x }, T = 0\n");
  wputsw(" Step 2 . If |L| = n then stop and output T.");
  wputs("\n");
  wputsw(" Step 3 . Else, find all edges with one vertex Ui in L and the"
      " other Vj which is not in L yet. Pick the one with least weight,"
      " (U, V)");
  wputs("\n      L <- L U {V}");
  wputs("\n      T <- T U {U, V}");
  wputs("\n      go to Step 2.");
  press_a_key(22);
  wslide(0,27);
  short_delay();
  alg_ex_prim_1();
}

/**********************************************************************/
/* This routine gives an example about six towns and high ways between these    */
/* towns. It shows the implementation of Prim's algorithm to this problem       */
/* in step by step basis.                                                       */
/**********************************************************************/
static void alg_ex_prim_1 (void)
{
  /********************************************************************/
  /* attach [Pageup] to the prim_alg() function */
  setonkey(0x4900,P4,0);
  /********************************************************************/
  /* attach [Pagedown] to the alg_ex_prim_2() function */
  setonkey(0x5100,P6,0);
  /********************************************************************/
  if((w[6]=wopen(6,15,12,65,3,GREEN|_BLACK,BLACK|_RED))==0)
          error_exit(1);
  wtitle("[Minimal Spanning Trees - Example_4_4_2]",
        TCENTER,_LGREY|LBLUE);
  add_shadow();
  wputsw(" Now let's go back to our first example and see how we are"
          " going to apply this algorithm step-by-step");
  press_a_key(4);
  wcloseall();
  spawnl(P_WAIT,"examp442.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  alg_ex_prim_2();
}
```

```
/**********************************************************************/
/* Another example about a Prim' Algorithm implementation            */
/**********************************************************************/
static void alg_ex_prim_2 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the alg_ex_prim_1() function */
  setonkey(0x4900,P5,0);
  /*******************************************************************/
  /* attach [Pagedown] to the kruskals_alg() function */
  setonkey(0x5100,P7,0);
  /*******************************************************************/
  if((w[7]=wopen(6,15,11,65,3,GREEN|_BLACK,BLACK|_RED))==0)
            error_exit(1);
  wtitle("[Minimal Spanning Trees - Example_4_4_3]",
          TCENTER,_LGREY|LBLUE);
  add_shadow();
  wputs("\n        How about one more example ?");
  press_a_key(3);
  wcloseall();
  spawnl(P_WAIT,"examp443.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  kruskals_alg();
}
```

```
/*********************************************************************/
/* routine that introduces the Kruskal's minimal spanning tree algorithm        */
/*********************************************************************/
static void kruskals_alg(void)
{
  /*******************************************************************/
  /* attach [Pageup] to the alg_ex_prim_2() function */
  setonkey(0x4900,P6,0);
  /* attach [Pagedown] to the ex_kruskal_1() function */
  setonkey(0x5100,P8,0);
  /*******************************************************************/
  if((w[8]=wopen(6,20,11,58,3,GREEN|_BLACK,BLACK|_RED))==0)
            error_exit(1);
  wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|LBLUE);
  add_shadow();
  wputs("\n  Second algorithm is called Kruskal's algorithm.");
  press_a_key(3);
  wslide(1,20);
  if((w[9]=wopen(6,15,18,65,3,BLACK|_GREEN,RED|_CYAN))==0) error_exit(1);
  wtitle("[Kruskal's Minimal Spanning Tree Algorithm]",
          TCENTER,BLUE|_LGREY);
  add_shadow();
  wputsw("The algorithm  is as follows :");
  wputs("\n\n");
  wputsw("  Step 1. Order the edges from smallest weight to largest.");
  wputs("\n");
  wputsw("  Step 2. Add the edges in order, as long as a cycle is not"
           " created. T can be disconnected until it's completed." );
  wputs("\n");
  wputsw("  Step 3. If all nodes are visited STOP, or else GO TO Step 2.");
  press_a_key(10);
  short_delay();
  wslide(7,15);
  short_delay();
  ex_kruskal_1();
}
```

```
/************************************************************************/
/* This routine gives an step by step example implementation of Kruskal's        */
/* algorithm                                                              */
/************************************************************************/
static void ex_kruskal_1 (void)
{
  /**********************************************************************/
  /* attach [Pageup] to the kruskals_alg() function */
  setonkey(0x4900,P7,0);
  /**********************************************************************/
  /* attach [Pagedown] to the ex_kruskal_2() function */
  setonkey(0x5100,P9,0);
  /**********************************************************************/
  if((w[10]=wopen(6,15,10,65,3,BLACK|_GREEN,BLACK|_LGREY))==0)
          error_exit(1);
  wtitle("[Kruskal's Algorithm - Example_4_4_4]",TCENTER,BLUE|_LGREY);
  add_shadow();
  wputs("\n      It is better to see an example...");
  press_a_key(2);
  wcloseall();
  spawnl(P_WAIT,"examp444.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  ex_kruskal_2();
}
```

```
/*********************************************************************/
/* This routine gives an step by step example implementation of Kruskal's    */
/* algorithm                                                                   */
/*********************************************************************/
static void ex_kruskal_2 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the ex_kruskal_1() function */
  setonkey(0x4900,P8,0);
  /*******************************************************************/
  /* attach [Pagedown] to the exercises() function */
  setonkey(0x5100,P10,0);
  /*******************************************************************/
  if((w[1]=wopen(6,15,11,65,3,GREEN|_BLACK,BLACK|_RED))==0)
          error_exit(1);
  wtitle("[Minimal Spanning Trees - Example_4_4_5]",
         TCENTER,_LGREY|LBLUE);
  add_shadow();
  wputsw("  We now will solve the second example that we solved"
          " with Prim's algorithm by using Kruskal's algorithm.");
  press_a_key(3);
  wcloseall();
  spawnl(P_WAIT,"examp445.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exercises();
}
```

```
/************************************************************************/
/* This routine makes a small quiz about the minimal spanning trees.        */
/************************************************************************/
void exercises(void)
 {
   register int *screen;

   /********************************************************************/
   /* attach [Pageup] to the ex_kruskal_2() function          */
   setonkey(0x4900,P9,0);
   /********************************************************************/
   /* attach [Pagedown] to the exer1() function */
   setonkey(0x5100,P11,0);
   /********************************************************************/
   if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
            error_exit(1);
   wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   add_shadow():
   wputs("\n");
   wputsw(" We have completed our presentation of this section. Are"
        " you ready for a pop quiz ? ");
   press_a_key(3);
   short_delay():
   wclose();
   if((screen=ssave())==NULL) error_exit(3); {
    exer1();
   /********************************************************************/
   /* if mouse exists, turn on full mouse support */
    if(msinit()) {
      mssupport(MS_FULL);
      msgotoxy(12,49);
      }
   }
   srestore(screen);
 }
```

```
/***********************************************************************/
/* Dummy function to call the actual exercise 4.4.1                   */
/***********************************************************************/
static void exer1(void)
{
  /*******************************************************************/
  /* attach [Pageup] to the ex_kruskal_2() function          */
  setonkey(0x4900,P9,0);
  /*******************************************************************/
  /* attach [Pagedown] to the exer2() function */
  setonkey(0x5100,P12,0);
  /*******************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the first question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q441.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer2();
}
```

```
/****************************************************************************/
/* Dummy function to call the actual exercise 4.4.2                        */
/****************************************************************************/
static void exer2(void)
{
  /************************************************************************/
  /* attach [Pageup] to the exer1() function          */
  setonkey(0x4900,P11,0);
  /************************************************************************/
  /* attach [Pagedown] to the exer3() function */
  setonkey(0x5100,P13,0);
  /************************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the second question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q442.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer3();
}
```

```
/***********************************************************************/
/* Dummy function to call the actual exercise 4.4.3                    */
/***********************************************************************/
static void exer3(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the exer2() function          */
  setonkey(0x4900,P12,0);
  /*********************************************************************/
  /* attach [Pagedown] to the exer4() function */
  setonkey(0x5100,P14,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the third question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q443.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer4();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.4.4                       */
/*************************************************************************/
static void exer4(void)
{
  /**********************************************************************/
  /* attach [Pageup] to the exer3() function          */
  setonkey(0x4900,P13,0);
  /**********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Minimal Spanning Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the forth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q444.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  normal_exit();
}
```

```
/* PROGRAM   : examp441.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for minimal spanning
                 trees.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>               /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;



/**********************************************************************/
/* This function is used for including drivers to the executable code */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /********************************************************************/
  settext();
  /********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      }
  forecolor = WHITE;
}
```

```
/**********************************************************************/
/* This function sets the text default values                         */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
      closegraph();
      videoinit();
      exit(0);
   }
  }



/**********************************************************************/
/* main routine  which calls exer routine                             */
/**********************************************************************/
void main()
{
  exer();
}
```

```
/**********************************************************************/
/* This routine illustrates a minimal spanning tree.                  */
/**********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-4-1");
  pieslice(25*x,4*y,0,359,2);        /* Marina      */
  pieslice(55*x,2*y,0,359,2);        /* Greenwillage */
  pieslice(65*x,8*y,0,359,2);        /* Bigsur      */
  pieslice(30*x,8*y,0,359,2);        /* Monterey    */
  pieslice(45*x,11*y/4,0,359,2);     /* Salinas     */
  pieslice(50*x,8*y,0,359,2);        /* Carmel      */
  moveto(25*x,4*y); lineto(55*x,2*y);  lineto(65*x,8*y);
  lineto(30*x,8*y); lineto(25*x,4*y);
  moveto(30*x,8*y); lineto(45*x,11*y/4); lineto(50*x,8*y);
  outtextxy(24*x,3*y,"Marina");
  outtextxy(40*x,5*y/3,"Salinas");
  outtextxy(57*x,2*y,"Greenwillage");
  outtextxy(67*x,8*y,"Big Sur");
  outtextxy(45*x,9*y,"Carmel");
  outtextxy(25*x,9*y,"Monterey");
  outtextxy(33*x,4*y,"8");
  outtextxy(49*x,3*y,"3");
  outtextxy(63*x,6*y,"30");
  outtextxy(56*x,15*y/2,"15");
  outtextxy(37*x,15*y/2,"15");
  outtextxy(37*x,6*y,"12");
  outtextxy(29*x,6*y,"5");
  outtextxy(50*x,6*y,"15");
  /*********************************************************************/
  outtextxy(2*x,13*y,"Here we have the town names as nodes and the roads as
                    edges of the graph.");
outtextxy(2*x,14*y,"The problem is to find a road network of minimal total length
                    that connects");
outtextxy(2*x,15*y,"all the towns.");
outtextxy(2*x,17*y,"By inspection, we can begin by including those roads between
                    that are seperated");
outtextxy(2*x,18*y,"by the least distance. There must be a path between any pair
                    of towns, but");
outtextxy(2*x,19*y,"there must not be any roads that would cause a loop to form,
                    since that leads");
outtextxy(2*x,20*y,"to extra paths. This leads us to a new concept,  MINIMAL
                    SPANNING TREE.");
outtextxy(2*x,21*y,"Now let's leave this problem at this stage and see some
                    definitions. Later on");
outtextxy(2*x,22*y,"we will come back and see how we will solve it.");
/******************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM   : examp442.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for implementation
                 of Prim's algorithm.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code  */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*****************************************************************************/
/* This fuction initializes the necessary graphical routines                */
/*****************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }
```

```
/**********************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
```

```
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*********************************************************************/
/* This function sets the text default values                       */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen           */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*********************************************************************/
/* main routine that calls exer routine                             */
/*********************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates an implementation of Prim's MST algorithm.        */
/*********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-4-2");
  /******************************************************************/
  pieslice(3*x,4*y,0,359,2);       /* Marina      */
  pieslice(33*x,2*y,0,359,2);      /* Greenwillage */
  pieslice(43*x,8*y,0,359,2);      /* Bigsur      */
  pieslice(8*x,8*y,0,359,2);       /* Monterey    */
  pieslice(23*x,11*y/4,0,359,2);   /* Salinas     */
  pieslice(28*x,8*y,0,359,2);      /* Carmel      */
  moveto(3*x,4*y); lineto(23*x,11*y/4); lineto(33*x,2*y);
  lineto(43*x,8*y); lineto(8*x,8*y);    lineto(3*x,4*y);
  moveto(8*x,8*y); lineto(23*x,11*y/4); lineto(28*x,8*y);
  outtextxy(2*x,3*y,"Marina");
  outtextxy(18*x,5*y/3,"Salinas");
  outtextxy(28*x,5*y/3,"Greenwillage");
  outtextxy(36*x,17*y/2,"Big Sur");
  outtextxy(23*x,9*y,"Carmel");
  outtextxy(3*x,9*y,"Monterey");
  outtextxy(11*x,4*y,"8");      /* (Marina, Salinas)      */
  outtextxy(27*x,3*y,"3");      /* (Salinas, Greenwillage) */
  outtextxy(36*x,6*y,"30");     /* (Greenwillage, Bigsur)  */
  outtextxy(34*x,15*y/2,"15"); /* (Carmel, Bigsur);       */
  outtextxy(15*x,15*y/2,"15"); /* (Monterey, Carmel)      */
  outtextxy(15*x,6*y,"12");     /* (Monterey, Salinas)     */
  outtextxy(7*x,6*y,"5");       /* (Monterey, Marina)      */
  outtextxy(28*x,6*y,"15");     /* (Salinas, Carmel)       */
  /******************************************************************/
  outtextxy(10*x,12*y,"L");
outtextxy(25*x,12*y,"EDGES TO CHECK");
outtextxy(52*x,12*y,"DISTANCE");
outtextxy(77*x,12*y,"T");
moveto(2*x,25*y/2);
lineto(18*x,25*y/2);
moveto(20*x,25*y/2);
lineto(49*x,25*y/2);
moveto(51*x,25*y/2);
lineto(61*x,25*y/2);
moveto(63*x,25*y/2);
lineto(88*x,25*y/2);
/*****************************************************************/
outtextxy(47*x,3*y/2,"THE WAY WE APPLIED PRIM'S ALG.");
moveto(43*x,2*y);
lineto(89*x,2*y);
outtextxy(43*x,3*y,". We arbitrarily chose Monterey and put ");
outtextxy(43*x,4*y,"  her in L.");
outtextxy(5*x,13*y,"Monterey");
outtextxy(43*x,5*y,". We listed all edges going out from Mon-");
outtextxy(43*x,6*y,"  terey and put them in edges to check.");
outtextxy(20*x,13*y,"(Monterey, Marina)");
outtextxy(56*x,13*y,"5");
outtextxy(20*x,14*y,"(Monterey, Salinas)");
outtextxy(55*x,14*y,"12");
outtextxy(20*x,15*y,"(Monterey, Carmel)");
outtextxy(55*x,15*y,"15");
outtextxy(43*x,7*y,". We chose (Monterey,Marina) since it has");
outtextxy(43*x,8*y,"  the minimum distance. And we delete");
outtextxy(43*x,9*y,"  this edge from the check list.");
outtextxy(63*x,13*y,"(Monterey, Marina)");
setcolor(backcolor);
moveto(8*x,8*y); lineto(3*x,4*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(8*x,8*y); lineto(3*x,4*y); /* add (Monterey, Marina) to T */
setlinestyle(0,0,3);
```

```
moveto(20*x,13*y); lineto(40*x,13*y);/*delete (Monterey,Marina) from list*/
Pause(55*x,10*y);
setcolor(backcolor);
bar(43*x,5*y/2,MaxX-3*x/2,45*y/4);
setcolor(forecolor);
/*********************************************************************/
outtextxy(43*x,3*y,". We now add Marina to L since  L  <=  ");
outtextxy(43*x,4*y,"  L  U  {V}");
outtextxy(5*x,16*y,"Marina");
outtextxy(43*x,5*y,". We listed all edges going out from Ma-");
outtextxy(43*x,6*y,"  rina and put them in edges to check.");
outtextxy(20*x,16*y,"(Marina, Salinas)");
outtextxy(56*x,16*y,"8");
outtextxy(43*x,7*y,". We chose (Marina, Salinas) since it has");
outtextxy(43*x,8*y,"  the minimum distance among the existing.");
outtextxy(43*x,9*y,"  edges. And we deleted this edge from the");
outtextxy(43*x,10*y,"  check list.");
outtextxy(63*x,16*y,"(Marina, Salinas)");
setcolor(backcolor);
moveto(3*x,4*y); lineto(23*x,11*y/4);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(3*x,4*y); lineto(23*x,11*y/4); /* add (Marina,Salinas) to T */
setlinestyle(0,0,3);
moveto(20*x,16*y); lineto(40*x,16*y);/*delete (Marina,Salinas) from list*/
Pause(55*x,11*y);
setcolor(backcolor);
bar(43*x,5*y/2,MaxX-3*x/2,45*y/4);
setcolor(forecolor);
/*********************************************************************/
outtextxy(43*x,3*y,". We now add Salinas to L since  L  <=  ");
outtextxy(43*x,4*y,"  L  U  {V}");
outtextxy(5*x,17*y,"Salinas");
outtextxy(43*x,5*y,". We listed all edges going out from Sa-");
outtextxy(43*x,6*y,"  linas and put them in edges to check.");
outtextxy(43*x,7*y,"  And we deleted (Monterey, Salinas) from");
outtextxy(43*x,8*y," the list because it would cause cycle.");
outtextxy(20*x,17*y,"(Salinas, Greenwillage)");
outtextxy(56*x,17*y,"3");
outtextxy(20*x,18*y,"(Salinas, Carmel)");
outtextxy(55*x,18*y,"15");
outtextxy(43*x,9*y,". We choose (Salinas, Greenwillage) and ");
outtextxy(43*x,10*y," delete this edge from the check list.");
outtextxy(63*x,17*y,"(Salinas, Greenwillage)");
setcolor(backcolor);
moveto(23*x,11*y/4); lineto(33*x,2*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(23*x,11*y/4); lineto(33*x,2*y); /* add (Salinas,Greenwillage to T*/
setlinestyle(0,0,3);
moveto(20*x,17*y); lineto(44*x,17*y);/*delete (Salinas,Greenwillage) from list*/
moveto(20*x,14*y); lineto(41*x,14*y);/*delete (Monterey,Salinas) from list*/
Pause(55*x,11*y);
setcolor(backcolor);
bar(43*x,5*y/2,MaxX-3*x/2,45*y/4);
setcolor(forecolor);
/*****************************************************************/
outtextxy(43*x,3*y,". We add Greenwillage to L ");
outtextxy(5*x,19*y,"Greenwillage");
outtextxy(43*x,4*y,". We listed all edges going out from ");
outtextxy(43*x,5*y," Greenwillage in the check list.");
outtextxy(20*x,19*y,"(Greenwillage, Bigsur)");
outtextxy(55*x,19*y,"30");
outtextxy(43*x,6*y,". We chose (Monterey, Carmel) since ");
outtextxy(43*x,7*y," it is one of the least distances ");
outtextxy(43*x,8*y," in the list. Here we could also ");
outtextxy(43*x,9*y," chose (Salinas, Carmel).");
outtextxy(63*x,15*y,"(Monterey, Carmel)");
setcolor(backcolor);
moveto(8*x,8*y); lineto(28*x,8*y);
setcolor(forecolor);
setlinestyle(3,0,3);
```

```
moveto(8*x,8*y); lineto(28*x,8*y);/* add (Monterey, Carmel) to T */
setlinestyle(0,0,3);
moveto(20*x,15*y); lineto(40*x,15*y);/*delete (Monterey,Carmel) from list*/
Pause(55*x,11*y);
setcolor(backcolor);
bar(43*x,5*y/2,MaxX-3*x/2,45*y/4);
setcolor(forecolor);
/***********************************************************************/
outtextxy(43*x,3*y,". This time we add Carmel to L  ");
outtextxy(5*x,20*y,"Carmel");
outtextxy(43*x,4*y,". We listed all edges going out from ");
outtextxy(43*x,5*y,"  Carmel in the check list. But we dele-");
outtextxy(43*x,6*y,"  ted (Salinas, Carmel) from the list ");
outtextxy(43*x,7*y,"  otherwise it would cause a cycle. ");
outtextxy(20*x,20*y,"(Carmel,Bigsur)");
outtextxy(55*x,20*y,"15");
outtextxy(43*x,8*y,". This time we chose (Carmel, Bigsur)");
outtextxy(43*x,9*y,"  since it has the least distance.  ");
outtextxy(63*x,20*y,"(Carmel, Bigsur)");
setcolor(backcolor);
moveto(28*x,8*y); lineto(43*x,8*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(28*x,8*y); lineto(43*x,8*y);/* add (Carmel, Bigsur) to T */
setlinestyle(0,0,3);
moveto(20*x,20*y); lineto(40*x,20*y);/*delete (Carmel,Bigsur) from list*/
moveto(20*x,18*y); lineto(40*x,18*y);/*delete (Salinas,Carmel) from list*/
Pause(55*x,11*y);
setcolor(backcolor);
bar(43*x,5*y/2,MaxX-3*x/2,45*y/4);
setcolor(forecolor);
/***********************************************************************/
outtextxy(43*x,3*y,". Finally we add Bigsur to L which ");
outtextxy(43*x,4*y,"  is the last vertice in our graph.");
outtextxy(43*x,5*y,"  As you realize we can not add any");
outtextxy(43*x,6*y,"  other edge to out tree T. That is");
```

```
    outtextxy(43*x,7*y,"  we are done.");
    outtextxy(5*x,21*y,"Bigsur");
    moveto(20*x,19*y); lineto(43*x,19*y);
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : examp443.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 18, 1990
   REVISED    : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for implementation
                 of prims algorithm.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                     /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                      /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions           */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions      */
static void exer            (void);


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;

/*************************************************************************/
/* This function is used for including drivers to the executable code   */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**********************************************************************/
/* This fuction initializes the necessary graphical routines         */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }
```

```
/************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)          {
     case 'y': closegraph();
            videoinit();
            exit(0);
            break;
     case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
     case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);    /* restore any hidden hot keys */
}
/**********************************************************************/
/* This function sets the text default values                         */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/**********************************************************************/
/* main routine  which calls exer routine                             */
/**********************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates an inplementation of the Prim's MST algorithgm.     */
/*********************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE 4-4-3");
   /*****************************************************************/
   pieslice(5*x,7*y,0,359,2);      /* A */
   pieslice(10*x,7*y,0,359,2);     /* B */
   pieslice(35*x,7*y,0,359,2);     /* G */
   pieslice(20*x,4*y,0,359,2);     /* D */
   pieslice(20*x,10*y,0,359,2);    /* E */
   pieslice(20*x,3*y/2,0,359,2);   /* C */
   pieslice(20*x,25*y/2,0,359,2);  /* F */
   outtextxy(3*x,7*y,"A");
   outtextxy(36*x,7*y,"G");
   outtextxy(20*x/2,15*y/2,"B");
   outtextxy(20*x,9*y/2,"D");
   outtextxy(21*x,3*y/2,"C");
   outtextxy(20*x,19*y/2,"E");
   outtextxy(21*x,25*y/2,"F");
   moveto(5*x,7*y); lineto(10*x,7*y); lineto(35*x,7*y);
   outtextxy(8*x,27*y/4,"1");     /* (A,B) */
   outtextxy(20*x,27*y/4,"5");    /* (B,G) */
   moveto(5*x,7*y); lineto(20*x,4*y);   lineto(10*x,7*y);
   moveto(5*x,7*y); lineto(20*x,3*y/2);  lineto(20*x,4*y);
   moveto(5*x,7*y); lineto(20*x,10*y);  lineto(10*x,7*y);
   moveto(5*x,7*y); lineto(20*x,25*y/2); lineto(20*x,10*y);
   moveto(20*x,3*y/2);  lineto(35*x,7*y);
   moveto(20*x,4*y);   lineto(35*x,7*y);
   moveto(20*x,10*y);  lineto(35*x,7*y);
   moveto(20*x,25*y/2); lineto(35*x,7*y);
```

```
outtextxy(21*x,3*y,"2");        /* (C,D) */
outtextxy(21*x,11*y,"2");        /* (E,F) */
outtextxy(9*x,9*y/2,"4");        /* (A,C) */
outtextxy(11*x,8*y,"2");        /* (A,E) */
outtextxy(14*x,8*y,"3");        /* (B,E) */
outtextxy(29*x/2,6*y,"3");       /* (B,D) */
outtextxy(21*x/2,11*y/2,"2");  /* (A,D) */
outtextxy(30*x,9*y/2,"4");       /* (C,G) */
outtextxy(25*x,17*y/2,"3");     /* (E,G) */
outtextxy(21*x/2,10*y,"3");     /* (A,F) */
outtextxy(27*x,21*y/2,"3");     /* (F,G) */
outtextxy(25*x,11*y/2,"3");     /* (D,G) */
/*****************************************************************/
outtextxy(46*x,2*y,"L");
outtextxy(52*x,2*y,"EDGES TO CHECK");
outtextxy(73*x,2*y,"DISTANCE");
outtextxy(86*x,2*y,"T");
moveto(44*x,5*y/2); lineto(49*x,5*y/2);
moveto(51*x,5*y/2); lineto(70*x,5*y/2);
moveto(72*x,5*y/2); lineto(82*x,5*y/2);
moveto(84*x,5*y/2); lineto(89*x,5*y/2);
outtextxy(2*x,14*y,"THE WAY WE APPLIED PRIM'S ALGORITHM");
moveto(3*x/2,29*y/2); lineto(43*x,29*y/2);
/*****************************************************************/
outtextxy(2*x,15*y,". We again arbitrarily chose A and ");
outtextxy(2*x,16*y,"  put her in L.");
outtextxy(46*x,3*y,"A");
outtextxy(2*x,17*y,". We list all edges going out from A");
outtextxy(2*x,18*y,"  and put them in edges to check. And,");
outtextxy(2*x,19*y,"  write their weights under DISTANCE");
outtextxy(58*x,3*y,"(A,C)");
outtextxy(76*x,3*y,"4");
outtextxy(58*x,4*y,"(A,D)");
outtextxy(76*x,4*y,"2");
outtextxy(58*x,5*y,"(A,B)");
outtextxy(76*x,5*y,"1");
outtextxy(58*x,6*y,"(A,E)");
outtextxy(76*x,6*y,"2");
outtextxy(58*x,7*y,"(A,F)");
outtextxy(76*x,7*y,"3");
outtextxy(2*x,20*y,". We chose (A,B) since it has the least");
outtextxy(2*x,21*y,"  distance and we delete this edge from");
outtextxy(2*x,22*y,"  the check list.");
outtextxy(84*x,5*y,"(A,B)");
setcolor(backcolor);
moveto(5*x,7*y);  lineto(10*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(5*x,7*y);  lineto(10*x,7*y); /* add (A, B) to T */
setlinestyle(0,0,3);
moveto(58*x,5*y); lineto(63*x,5*y); /* delete (A,B) from the list*/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,50*x,24*y);
setcolor(forecolor);
/*************************************************************************/
outtextxy(2*x,15*y,". We now add B to L since  ");
outtextxy(2*x,16*y,"    L <= L  U  {V}");
outtextxy(46*x,8*y,"B");
outtextxy(2*x,17*y,". We listed all edges going out from B");
outtextxy(2*x,18*y,"  and put them in the check list.");
outtextxy(58*x,8*y,"(B,D)");
outtextxy(76*x,8*y,"3");
outtextxy(58*x,9*y,"(B,G)");
outtextxy(76*x,9*y,"5");
outtextxy(58*x,10*y,"(B,E)");
outtextxy(76*x,10*y,"3");
outtextxy(2*x,19*y,". We chose (A,D) since it has the least dist-");
outtextxy(2*x,20*y,"  ance among the existing edges. And we delet-");
outtextxy(2*x,21*y,"  ed this edge and (B,D) from the check list.");
outtextxy(2*x,22*y,"  (if we chose (B,D) we would have a cycle)");
outtextxy(84*x,4*y,"(A,D)");
```

```
setcolor(backcolor);
moveto(20*x,4*y);  lineto(5*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,4*y);  lineto(5*x,7*y);  /* add (A,D) to T */
setlinestyle(0,0,3);
moveto(58*x,4*y);  lineto(63*x,4*y); /* delete (A,D) from the list */
moveto(58*x,8*y);  lineto(77*x,8*y); /* delete (B,D) from the list */
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
setcolor(forecolor);
/*******************************************************************/
outtextxy(2*x,15*y,". We now add D to L.");
outtextxy(46*x,11*y,"D");
outtextxy(2*x,16*y,". We listed all edges going out from D");
outtextxy(2*x,17*y,"  and put them in the check list.");
outtextxy(58*x,11*y,"(D,C)");
outtextxy(76*x,11*y,"2");
outtextxy(58*x,12*y,"(D,G)");
outtextxy(76*x,12*y,"3");
outtextxy(2*x,18*y,". We chose (A,E) since it has the least dist-");
outtextxy(2*x,19*y,"  ance among the existing edges. And we delet-");
outtextxy(2*x,20*y,"  ed (A,E) and (B,E) from the check list. (It");
outtextxy(2*x,21*y,"  would cause cycle if chose (B,E)). Here we");
outtextxy(2*x,22*y,"  could choose (D,C) also but (A,E) is the");
outtextxy(2*x,23*y,"  first one, that's why we chose (A,E).");
outtextxy(84*x,6*y,"(A,E)");
setcolor(backcolor);
moveto(20*x,10*y);  lineto(5*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,10*y);  lineto(5*x,7*y);  /* add (A,E) to T */
setlinestyle(0,0,3);
moveto(58*x,6*y);  lineto(63*x,6*y); /* delete (A,E) from the list */
moveto(58*x,10*y);  lineto(77*x,10*y);/* delete (B,E) from the list */
```

```
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
setcolor(forecolor);
/***********************************************************************/
outtextxy(2*x,15*y,". We now add E to L.");
outtextxy(46*x,13*y,"E");
outtextxy(2*x,16*y,". We listed all edges going out from E");
outtextxy(2*x,17*y,"  and put them in the check list.");
outtextxy(58*x,13*y,"(E,F)");
outtextxy(76*x,13*y,"2");
outtextxy(58*x,14*y,"(E,G)");
outtextxy(76*x,14*y,"3");
outtextxy(2*x,18*y,". We chose (D,C) since it has the least dist-");
outtextxy(2*x,19*y,"  ance among the existing edges. And we delet-");
outtextxy(2*x,20*y,"  ed (D,C) and (A,C) from the check list." );
outtextxy(2*x,21*y,"  (It would cause a cycle if we chose (A,C)).");
outtextxy(84*x,11*y,"(D,C)");
setcolor(backcolor);
moveto(20*x,4*y);  lineto(20*x,3*y/2);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,4*y);  lineto(20*x,3*y/2);  /* add (D,C) to T */
setlinestyle(0,0,3);
moveto(58*x,11*y); lineto(63*x,11*y);  /* delete (D,C) from the list */
moveto(58*x,3*y);  lineto(77*x,3*y);   /* delete (A,C) from the list */
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
setcolor(forecolor);
/***********************************************************************/
outtextxy(2*x,15*y,". We now add C to L.");
outtextxy(46*x,15*y,"C");
outtextxy(2*x,16*y,". We listed all edges going out from");
outtextxy(2*x,17*y,"  (i.e. (C,G) ) and put them in the");
outtextxy(2*x,18*y,"  check list.");
outtextxy(58*x,15*y,"(C,G)");
outtextxy(76*x,15*y,"4");
outtextxy(2*x,19*y,". We chose (E,F) since it has the");
outtextxy(2*x,20*y," least distance among the existing");
outtextxy(2*x,21*y," edges. And we deleted (E,F) and ");
outtextxy(2*x,22*y," (A,F) from the check list.(It would");
outtextxy(2*x,23*y," cause a cycle if we chose (A,F)).");
outtextxy(84*x,13*y,"(E,F)");
setcolor(backcolor);
moveto(20*x,10*y); lineto(20*x,25*y/2);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,10*y); lineto(20*x,25*y/2); /* add (E,F) to T */
setlinestyle(0,0,3);
moveto(58*x,13*y); lineto(63*x,13*y); /* delete (E,F) from the list */
moveto(58*x,7*y); lineto(77*x,7*y); /* delete (A,F) from the list */
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,45*x,47*y/2);
bar(3*x/2,23*y,70*x,49*y/2);
setcolor(forecolor);
/*************************************************************************/
outtextxy(2*x,15*y,". We now add F to L.");
outtextxy(46*x,16*y,"F");
outtextxy(2*x,16*y,". We listed all edges going out from");
outtextxy(2*x,17*y," (i.e. (F,G) ) and put them in the");
outtextxy(2*x,18*y," check list.");
outtextxy(58*x,16*y,"(F,G)");
outtextxy(76*x,16*y,"3");
outtextxy(2*x,19*y,". We chose (D,G) since it has the");
outtextxy(2*x,20*y," least distance among the existing");
outtextxy(2*x,21*y," edges. And we deleted (D,G) and");
outtextxy(2*x,22*y," (B,G),(E,G),(C,G),(F,G) from the");
outtextxy(2*x,23*y," check list.(We would have cycle");
outtextxy(2*x,47*y/2," if we chose any one of them).");
outtextxy(84*x,12*y,"(D,G)");
```

```
  setcolor(backcolor);
  moveto(20*x,4*y);  lineto(35*x,7*y);
  setcolor(forecolor);
  setlinestyle(3,0,3);
  moveto(20*x,4*y);  lineto(35*x,7*y);  /* add (D,G) to T */
  setlinestyle(0,0,3);
  moveto(58*x,12*y); lineto(63*x,12*y); /* delete (D,G) from the list */
  moveto(58*x,9*y);  lineto(77*x,9*y);  /* delete (B,G) from the list */
  moveto(58*x,14*y); lineto(63*x,14*y); /* delete (E,G) from the list */
  moveto(58*x,15*y); lineto(77*x,15*y); /* delete (C,G) from the list */
  moveto(58*x,16*y); lineto(77*x,16*y); /* delete (F,G) from the list */
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(3*x/2,59*y/4,45*x,49*y/2);
  bar(3*x/2,23*y,70*x 49*y/2);
  setcolor(forecolor);
  /**********************************************************************/
  outtextxy(2*x,15*y,". We finally  add G to L.");
  outtextxy(46*x,17*y,"G");
  outtextxy(2*x,16*y,". We see that there is no edge to");
  outtextxy(2*x,17*y,"  put in the check list. So, this");
  outtextxy(2*x,18*y,"  means we are done.");
  /**********************************************************************/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
}
```

```
/* PROGRAM   : examp444.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 18, 1990
   REVISED    : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for kruskal's
                 algorithm.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__ )                    /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/**********************************************************************/
/* This function is used for including drivers to the executable code */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/**************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
   }
   switch (ch)          {
    case 'y': closegraph();
           videoinit();
           exit(0);
           break;
    case 'Y': closegraph();
           videoinit();
           exit(0);
           break;
    case 'n': setcolor(backcolor);
           bar(4*x/3,23*y,30*x,97*y/4);
           bar(31*x,23*y,69*x,97*y/4);
           setcolor(forecolor);
           break;
    case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/**********************************************************************/
/* This function sets the text default values                         */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/**********************************************************************/
/* main routine  which calls exer routine                             */
/**********************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine illustrates an implementation of Kruskal's MST algorithm.     */
/*************************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-4-5");
  /***********************************************************************/
  pieslice(3*x,4*y,0,359,2);    /* Marina      */
  pieslice(33*x,2*y,0,359,2);   /* Greenwillage */
  pieslice(43*x,8*y,0,359,2);   /* Bigsur      */
  pieslice(8*x,8*y,0,359,2);    /* Monterey    */
  pieslice(23*x,11*y/4,0,359,2); /* Salinas    */
  pieslice(28*x,8*y,0,359,2);   /* Carmel      */
  moveto(3*x,4*y); lineto(33*x,2*y); lineto(43*x,8*y); lineto(8*x,8*y);
  lineto(3*x,4*y);
  moveto(8*x,8*y); lineto(23*x,11*y/4); lineto(28*x,8*y);
  outtextxy(2*x,3*y,"Marina");
  outtextxy(18*x,5*y/2,"Salinas");
  outtextxy(28*x,5*y/3,"Greenwillage");
  outtextxy(36*x,17*y/2,"Big Sur");
  outtextxy(23*x,9*y,"Carmel");
  outtextxy(3*x,9*y,"Monterey");
  outtextxy(11*x,4*y,"8");
  outtextxy(27*x,3*y,"3");
  outtextxy(36*x,6*y,"30");
  outtextxy(34*x,15*y/2,"15");
  outtextxy(15*x,15*y/2,"15");
  outtextxy(15*x,6*y,"12");
  outtextxy(7*x,6*y,"5");
  outtextxy(28*x,6*y,"15");
  /***********************************************************************/
  outtextxy(5*x,12*y,"EDGES TO CHECK");
outtextxy(32*x,12*y,"DISTANCE");
moveto(2*x,25*y/2);  lineto(29*x,25*y/2);
moveto(31*x,25*y/2); lineto(41*x,25*y/2);
outtextxy(44*x,12*y,"THE WAY WE APPLIED KRUSKAL'S ALG.");
moveto(43*x,25*y/2); lineto(90*x,25*y/2);
/*******************************************************************/
outtextxy(43*x,13*y,". We sorted the edges from least distance");
outtextxy(43*x,14*y,"  and listed them.");
outtextxy(2*x,13*y,"(Salinas, Greenwillage)");
outtextxy(36*x,13*y,"3");
outtextxy(2*x,14*y,"(Monterey, Marina)");
outtextxy(36*x,14*y,"5");
outtextxy(2*x,15*y,"(Marina, Salinas)");
outtextxy(36*x,15*y,"8");
outtextxy(2*x,16*y,"(Monterey, Salinas)");
outtextxy(35*x,16*y,"12");
outtextxy(2*x,17*y,"(Monterey, Carmel)");
outtextxy(35*x,17*y,"15");
outtextxy(2*x,18*y,"(Salinas, Carmel)");
outtextxy(35*x,18*y,"15");
outtextxy(2*x,19*y,"(Carmel, Bigsur)");
outtextxy(35*x,19*y,"15");
outtextxy(2*x,20*y,"(Bigsur, Greenwillage)");
outtextxy(35*x,20*y,"30");
/*******************************************************************/
outtextxy(43*x,15*y,". Now we will start to build the tree");
outtextxy(43*x,16*y,"  starting from the least edge.");
Pause(55*x,24*y);
bar(43*x,51*y/4,90*x,49*y/2);
/*******************************************************************/
/* Second graph having only the nodes. We will use this graph to show   */
/* how the minimal spanning tree grows on.                               */
/*******************************************************************/
pieslice(46*x,4*y,0,359,2);    /* Marina       */
pieslice(76*x,2*y,0,359,2);    /* Greenwillage */
pieslice(86*x,8*y,0,359,2);    /* Bigsur       */
pieslice(51*x,8*y,0,359,2);       /* Monterey    */
pieslice(66*x,11*y/4,0,359,2);  /* Salinas      */
pieslice(71*x,8*y,0,359,2);       /* Carmel       */
outtextxy(45*x,3*y,"Marina");
outtextxy(56*x,5*y/2,"Salinas");
outtextxy(71*x,5*y/3,"Greenwillage");
outtextxy(80*x,17*y/2,"Big Sur");
outtextxy(66*x,9*y,"Carmel");
outtextxy(46*x,9*y,"Monterey");
/**********************************************************************/
outtextxy(43*x,13*y,". As you will see we will start from");
outtextxy(43*x,14*y,"  first edge (the least distance) ");
outtextxy(43*x,15*y,"  and will connect the nodes in the");
outtextxy(43*x,16*y,"  next graph accordingly. ");
moveto(66*x,11*y/4); lineto(76*x,2*y);/* add (Salinas,Greenwillage) to tree*/
moveto(2*x,13*y); lineto(29*x,13*y);  /* delete this from the list        */
outtextxy(70*x,3*y,"3");
Pause(55*x,24*y);
bar(43*x,51*y/4,90*x,49*y/2);
/**********************************************************************/
moveto(51*x,8*y); lineto(46*x,4*y);   /* add (Monterey, Marina) to tree */
moveto(2*x,14*y); lineto(29*x,14*y);  /* delete this edge from the list */
outtextxy(47*x,6*y,"5");
Pause(55*x,24*y);
bar(43*x,51*y/4,90*x,49*y/2);
/**********************************************************************/
moveto(46*x,4*y); lineto(66*x,11*y/4); /* add (Marina,Salinas) to  tree */
moveto(2*x,15*y); lineto(29*x,15*y);   /* delete this edge from the list*/
outtextxy(56*x,4*y,"8");
outtextxy(43*x,16*y,"  Here as you see we cannot choose");
outtextxy(43*x,17*y,"  (Monterey, Salinas) because it");
outtextxy(43*x,18*y,"  cause a cycle. So we skip this edge.");
moveto(2*x,16*y); lineto(41*x,16*y);  /* delete (Monterey,Salinas) from the list*/
Pause(55*x,24*y);
bar(43*x,51*y/4,90*x,49*y/2);
/**********************************************************************/
```

```
    outtextxy(43*x,17*y,". Now we will continue from where ");
    outtextxy(43*x,18*y,"  we left, that is from (Monterey,");
    outtextxy(43*x,19*y,"  Carmel).");
    moveto(51*x,8*y); lineto(71*x,8*y);/* add (Monterey,Carmel) to tree   */
    moveto(2*x,17*y); lineto(29*x,17*y);/* delete this edge from the list */
    outtextxy(58*x,15*y/2,"15");
    Pause(55*x,24*y);
    bar(43*x,51*y/4,90*x,49*y/2);
    /*****************************************************************/
    outtextxy(43*x,18*y,". Again, here we cannot choose");
    outtextxy(43*x,19*y,"  (Salinas, Carmel) because of");
    outtextxy(43*x,20*y,"  cycle. So we skip to the next");
    outtextxy(43*x,21*y,"  edge, (Carmel, Bigsur).");
    moveto(2*x,18*y); lineto(41*x,18*y);/* delete (Carmel,Bigsur) from the list*/
    Pause(55*x,24*y);
    moveto(71*x,8*y); lineto(86*x,8*y);/* delete (Salinas,Carmel) from list */
    moveto(2*x,19*y); lineto(29*x,19*y);/* delete (Carmel,Bigsur) from list */
    outtextxy(77*x,15*y/2,"15");
    Pause(55*x,24*y);
    bar(43*x,51*y/4,90*x,49*y/2);
    /*****************************************************************/
    outtextxy(43*x,20*y,". Here, as you see we connected");
    outtextxy(43*x,21*y,"  all the existing edges. So we");
    outtextxy(43*x,22*y,"  are done.");
    /*****************************************************************/
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : examp445.c
  AUTHOR     : Atilla BAKAN
  DATE       : Apr. 18, 1990
  REVISED    : Apr. 18, 1990

  DESCRIPTION : This routine draws the example graph for a Kruskal's
                algorithm implementation.


  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                      /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b;
  #define findnext(a)     _dos_findnext(a)
  #define ffblk          find_t
  #define ff_name         name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name          name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/**********************************************************************/
/* This function is used for including drivers to the executable code */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }
```

```
/***********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(43*x,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(44*x,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      setcolor(backcolor);
      bar(43*x,23*y,179*x/2,97*y/4);
      setcolor(quitcolor);
      outtextxy(44*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(43*x,23*y,179*x/2,97*y/4);
      setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(43*x,23*y,179*x/2,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(43*x,23*y,179*x/2,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);    /* restore any hidden hot keys */
}


/************************************************************************/
/* This function sets the text default values                          */
/************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}


/************************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
```

```
/*********************************************************************/
/* main routine which calls exer routine                             */
/*********************************************************************/
void main()
{
  exer();
}


/*********************************************************************/
/* This routine illustrates an implementation of the Kruskal's MST algorithm.      */
/*********************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE 4-4-6");
   /*****************************************************************/
   pieslice(5*x,7*y,0,359,2);      /* A */
   pieslice(10*x,7*y,0,359,2);     /* B */
   pieslice(35*x,7*y,0,359,2);     /* G */
   pieslice(20*x,4*y,0,359,2);     /* D */
   pieslice(20*x,10*y,0,359,2);    /* E */
   pieslice(20*x,3*y/2,0,359,2);   /* C */
   pieslice(20*x,25*y/2,0,359,2);  /* F */
   outtextxy(3*x,7*y,"A");
   outtextxy(36*x,7*y,"G");
   outtextxy(20*x/2,15*y/2,"B");
   outtextxy(20*x,9*y/2,"D");
   outtextxy(21*x,3*y/2,"C");
   outtextxy(20*x,19*y/2,"E");
   outtextxy(21*x,25*y/2,"F");
   moveto(5*x,7*y); lineto(10*x,7*y); lineto(35*x,7*y);
   outtextxy(8*x,27*y/4,"1");      /* (A,B) */
outtextxy(20*x,27*y/4,"5");    /* (B,G) */
moveto(5*x,7*y); lineto(20*x,4*y);   lineto(10*x,7*y);
moveto(5*x,7*y); lineto(20*x,3*y/2);  lineto(20*x,4*y);
moveto(5*x,7*y); lineto(20*x,10*y);  lineto(10*x,7*y);
moveto(5*x,7*y); lineto(20*x,25*y/2); lineto(20*x,10*y);
moveto(20*x,3*y/2);  lineto(35*x,7*y);
moveto(20*x,4*y);   lineto(35*x,7*y);
moveto(20*x,10*y);  lineto(35*x,7*y);
moveto(20*x,25*y/2); lineto(35*x,7*y);
outtextxy(21*x,3*y,"2");       /* (C,D) */
outtextxy(21*x,11*y,"2");      /* (E,F) */
outtextxy(9*x,9*y/2,"4");      /* (A,C) */
outtextxy(11*x,8*y,"2");       /* (A,E) */
outtextxy(14*x,8*y,"3");       /* (B,E) */
outtextxy(29*x/2,6*y,"3");     /* (B,D) */
outtextxy(21*x/2,11*y/2,"2");  /* (A,D) */
outtextxy(30*x,9*y/2,"4");     /* (C,G) */
outtextxy(25*x,17*y/2,"3");    /* (E,G) */
outtextxy(21*x/2,10*y,"3");    /* (A,F) */
outtextxy(27*x,21*y/2,"3");    /* (F,G) */
outtextxy(25*x,11*y/2,"3");    /* (D,G) */
/*****************************************************************/
outtextxy(45*x,3*y/2,"EDGES TO CHECK");
outtextxy(67*x,3*y/2,"WEIGHT");
moveto(44*x,2*y); lineto(60*x,2*y);
moveto(65*x,2*y); lineto(75*x,2*y);
/*****************************************************************/
outtextxy(44*x,35*y/2,"THE WAY WE APPLIED KRUSKAL'S ALGORITHM");
moveto(44*x,18*y); lineto(90*x,18*y);
outtextxy(44*x,19*y,". To apply this algorithm we will sort ");
outtextxy(44*x,20*y,"  the edges from the least to the great-");
outtextxy(44*x,21*y,"  est and we will list them.");
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*****************************************************************/
outtextxy(49*x,3*y,"(A,B)"); outtextxy(70*x,3*y,"1");
outtextxy(49*x,4*y,"(A,D)");  outtextxy(70*x,4*y,"2");
outtextxy(49*x,5*y,"(A,E)");  outtextxy(70*x,5*y,"2");
outtextxy(49*x,6*y,"(C,D)");  outtextxy(70*x,6*y,"2");
outtextxy(49*x,7*y,"(E,F)");  outtextxy(70*x,7*y,"2");
outtextxy(49*x,8*y,"(A,F)");  outtextxy(70*x,8*y,"3");
outtextxy(49*x,9*y,"(B,D)");  outtextxy(70*x,9*y,"3");
outtextxy(49*x,10*y,"(B,E)"); outtextxy(70*x,10*y,"3");
outtextxy(49*x,11*y,"(D,G)"); outtextxy(70*x,11*y,"2");
outtextxy(49*x,12*y,"(E,G)"); outtextxy(70*x,12*y,"3");
outtextxy(49*x,13*y,"(F,G)"); outtextxy(70*x,13*y,"3");
outtextxy(49*x,14*y,"(A,C)"); outtextxy(70*x,14*y,"4");
outtextxy(49*x,15*y,"(C,G)"); outtextxy(70*x,15*y,"4");
outtextxy(49*x,16*y,"(B,G)"); outtextxy(70*x,16*y,"5");
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*********************************************************************/
outtextxy(44*x,19*y,". Now we are going to build the tree");
outtextxy(44*x,20*y,"  starting from the least edge.");
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*********************************************************************/
pieslice(5*x,19*y,0,359,2);      /* A */
pieslice(10*x,19*y,0,359,2);     /* B */
pieslice(35*x,19*y,0,359,2);     /* G */
pieslice(20*x,16*y,0,359,2);     /* D */
pieslice(20*x,22*y,0,359,2);     /* E */
pieslice(20*x,27*y/2,0,359,2);   /* C */
pieslice(20*x,49*y/2,0,359,2);   /* F */
outtextxy(3*x,19*y,"A");
outtextxy(36*x,19*y,"G");
outtextxy(20*x/2,39*y/2,"B");
outtextxy(20*x,33*y/2,"D");
outtextxy(21*x,13*y,"C");
outtextxy(20*x,43*y/2,"E");
outtextxy(21*x,49*y/2,"F");
/*********************************************************************/
```

```
moveto(5*x,19*y); lineto(10*x,19*y);
moveto(49*x,3*y); lineto(54*x,3*y);
outtextxy(8*x,75*y/4,"1");     /* (A,B) */
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*****************************************************************/
moveto(5*x,19*y); lineto(20*x,16*y);
moveto(49*x,4*y); lineto(54*x,4*y);
outtextxy(21*x/2,35*y/2,"2");  /* (A,D) */
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*****************************************************************/
moveto(5*x,19*y); lineto(20*x,22*y);
moveto(49*x,5*y); lineto(54*x,5*y);
outtextxy(11*x,20*y,"2");      /* (A,E) */
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*****************************************************************/
moveto(20*x,16*y); lineto(20*x,27*y/2);
moveto(49*x,6*y); lineto(54*x,6*y);
outtextxy(21*x,15*y,"2");      /* (C,D) */
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*****************************************************************/
moveto(20*x,22*y); lineto(20*x,49*y/2);
moveto(49*x,7*y); lineto(54*x,7*y);
outtextxy(21*x,23*y,"2");      /* (E,F) */
Pause(55*x,24*y);
bar(44*x,37*y/2,89*x,49*y/2);
/*****************************************************************/
outtextxy(44*x,19*y,". At this point we cannot add (A,F),");
outtextxy(44*x,20*y,"  (B,D), or (B,E) to the tree, because");
outtextxy(44*x,21*y,"  otherwise we would have cycle. So we");
outtextxy(44*x,22*y,"  we will skip these edges.");
moveto(49*x,8*y); lineto(72*x,8*y);
moveto(49*x,9*y); lineto(72*x,9*y);
```

```
    moveto(49*x,10*y); lineto(72*x,10*y);
    Pause(55*x,24*y);
    bar(44*x,37*y/2,89*x,49*y/2);
    /**********************************************************************/
    outtextxy(44*x,19*y,". We now continue from where we left.");
    moveto(20*x,16*y); lineto(35*x,19*y);
    moveto(49*x,11*y); lineto(54*x,11*y);
    outtextxy(25*x,73*y/4,"3");     /* (D,G) */
    outtextxy(44*x,20*y,"  We added  (D,G) to the tree. As you ");
    outtextxy(44*x,21*y,"  see adding this edge completed the");
    outtextxy(44*x,22*y,"  existing nodes in the tree. This means");
    outtextxy(44*x,23*y,"  we are done and we stop here.");
    /**********************************************************************/
    Pause(55*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : q441.c
  AUTHOR     : Atilla BAKAN
  DATE       : Mar. 22, 1990
  REVISED    : Apr. 22, 1990

  DESCRIPTION : This program contains the first exercise about the minimal
                spanning trees.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                       /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                       /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)       _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)      _dos_findnext(a)
  #define ffblk            find_t
  #define ff_name          name
#elif defined(__ZTC__)                         /* Zortech C/C++ */
  #define ffblk            FIND
  #define ff_name          name
  #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/*************************************************************************/
/* miscellaneous global variables                                          */
/*************************************************************************/
 int in_the_exercise = 1;


/*************************************************************************/
/* graphic initialization variables                                         */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code          */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}


/*********************************************************************/
/* This fuction initializes the necessary graphical routines                   */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*********************************************************************/
  settext();
 /*********************************************************************/
```

```
if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
   ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
    setfillstyle(SOLID_FILL,BLACK);
    backcolor = BLACK;
    quitcolor = WHITE;
    }
 else {
    setfillstyle(SOLID_FILL,BLUE);
    backcolor = BLUE;
    quitcolor = RED;
    }
forecolor = WHITE;
}


/***********************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL); /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
   }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   default : break;
   }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}


/***********************************************************************/
/* This function sets the text default values                         */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen           */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine that calls exer routine                              */
/*-----------------------------------------------------------------
-----*/
void main()
{
  exer();
}

/*********************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                      */
/*********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  1");
   /*****************************************************************/
   outtextxy(2*x,2*y,"Use Prim's algorithm to find a minimal spanning tree. (Start at
                        A. If there");
outtextxy(2*x,3*y,"is a choice of edges select edges according to alphabetical
                order.)");
/**********************************************************************/
pieslice(25*x,5*y,0,359,2);     /* A */
pieslice(35*x,5*y,0,359,2);     /* B */
pieslice(45*x,5*y,0,359,2);     /* C */
pieslice(55*x,5*y,0,359,2);     /* D */
pieslice(25*x,8*y,0,359,2);     /* E */
pieslice(35*x,8*y,0,359,2);     /* F */
pieslice(45*x,8*y,0,359,2);     /* G */
pieslice(55*x,8*y,0,359,2);     /* H */
pieslice(35*x,11*y,0,359,2);     /* I */
pieslice(45*x,11*y,0,359,2);     /* J */
/**********************************************************************/
outtextxy(25*x,9*y/2,"A");
outtextxy(35*x,9*y/2,"B");
outtextxy(45*x,9*y/2,"C");
outtextxy(55*x,9*y/2,"D");
outtextxy(23*x,8*y,"E");
outtextxy(33*x,17*y/2,"F");
outtextxy(47*x,17*y/2,"G");
outtextxy(57*x,8*y,"H");
outtextxy(35*x,23*y/2,"I");
outtextxy(45*x,23*y/2,"J");
/**********************************************************************/
outtextxy(30*x,9*y/2,"4");        /* (A, B) */
outtextxy(50*x,9*y/2,"3");        /* (C, D) */
outtextxy(23*x,13*y/2,"3");       /* (A, E) */
outtextxy(36*x,13*y/2,"5");       /* (B, F) */
outtextxy(43*x,13*y/2,"2");       /* (C, G) */
outtextxy(56*x,13*y/2,"2");       /* (D, H) */
outtextxy(50*x,6*y,"2");          /* (C, H) */
outtextxy(30*x,15*y/2,"6");       /* (E, F) */
outtextxy(40*x,15*y/2,"1");       /* (F, G) */
outtextxy(49*x,15*y/2,"4");       /* (G, H) */
outtextxy(33*x,19*y/2,"4");        /* (F, I) */
outtextxy(40*x,23*y/2,"3");        /* (I, J) */
outtextxy(46*x,19*y/2,"1");        /* (G, J) */
outtextxy(38*x,19*y/2,"2");        /* (G, I) */
moveto(25*x,5*y); lineto(35*x,5*y); lineto(35*x,8*y);
lineto(25*x,8*y); lineto(25*x,5*y);
moveto(45*x,8*y); lineto(45*x,11*y);lineto(35*x,11*y);
lineto(35*x,8*y); lineto(45*x,8*y);
moveto(35*x,11*y);lineto(45*x,8*y);
moveto(45*x,8*y); lineto(45*x,5*y); lineto(55*x,5*y);
lineto(55*x,8*y);lineto(45*x,8*y);
moveto(45*x,5*y);lineto(55*x,8*y);
/******************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,19*y,"   Please type a, b, c or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
   setcolor(backcolor);
   bar(50*x,37*y/2,88*x,20*y);
   setcolor(forecolor);
   }
  }
   switch (Ch)          {
   case 'a': outtextxy(47*x,19*y,"a");
     outtextxy(52*x,19*y,"You want to see the algorithm ");
     outtextxy(52*x,20*y,"again. Press any key to continue.");
     Pause(30*x,24*y);
```

```
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,21*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      show_alg();
      break;
    case 'b': outtextxy(47*x,19*y,"b");
      outtextxy(52*x,19*y,"You want to compare your solu-");
      outtextxy(52*x,20*y,"tion with ours. So press any  ");
      outtextxy(52*x,21*y,"key to see it.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,22*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      compare_solutions();
      break;
    case 'c': outtextxy(47*x,19*y,"c");
      outtextxy(52*x,19*y,"You want to see step by step");
      outtextxy(52*x,20*y,"solution. So press any key to ");
      outtextxy(52*x,21*y,"continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,22*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      step_solution();
      break;
    case 'd': outtextxy(47*x,19*y,"d");
      confirm_exit();
      break;
    default  : break;
   }
 }
  closegraph();
}
```

```
/**************************************************************************/
/* This routine gives Prim's minimal spanning tree algorithm             */
/**************************************************************************/
static void show_alg(void)
{
  outtextxy(5*x,15*y,"Step 1 . Pick an arbitrary initial vertex x.");
  outtextxy(5*x,16*y,"         L = { x }, T = 0");
  outtextxy(5*x,17*y,"Step 2 . If |L| = n then stop and output T.");
  outtextxy(5*x,18*y,"Step 3 . Else, find all edges with one vertex Ui in L and the
                    other Vj ");
  outtextxy(5*x,19*y,"        which is not in L yet. Pick the one with least weight,
                   (U, V)");
  outtextxy(5*x,20*y,"         L <- L U {V}");
  outtextxy(5*x,21*y,"         T <- T U {U, V}");
  outtextxy(5*x,22*y,"         go to Step 2.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,13*y,88*x,49*y/2);
  setcolor(forecolor);
}


/**************************************************************************/
/* This routine gives the solution to the exercise to be compared.       */
/**************************************************************************/
static void compare_solutions(void)
{
  setcolor(backcolor);        /* Clean the game field */
  bar(2*x,13*y,179*x/2,49*y/2);
  /**********************************************************************/
  moveto(25*x,5*y); lineto(35*x,5*y);
  moveto(25*x,5*y); lineto(25*x,8*y);
  moveto(35*x,5*y); lineto(35*x,8*y);
  moveto(35*x,8*y); lineto(45*x,8*y);
  moveto(45*x,8*y); lineto(35*x,11*y);
  moveto(45*x,8*y); lineto(45*x,11*y);
moveto(45*x,8*y); lineto(45*x,5*y);
moveto(45*x,5*y); lineto(55*x,8*y);
moveto(55*x,8*y); lineto(55*x,5*y);
/*******************************************************************/
setcolor(forecolor);
setlinestyle(3,0,3);
/*******************************************************************/
moveto(25*x,5*y); lineto(35*x,5*y);
moveto(25*x,5*y); lineto(25*x,8*y);
moveto(35*x,5*y); lineto(35*x,8*y);
moveto(35*x,8*y); lineto(45*x,8*y);
moveto(45*x,8*y); lineto(35*x,11*y);
moveto(45*x,8*y); lineto(45*x,11*y);
moveto(45*x,8*y); lineto(45*x,5*y);
moveto(45*x,5*y); lineto(55*x,8*y);
moveto(55*x,8*y); lineto(55*x,5*y);
/*******************************************************************/
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(25*x,5*y); lineto(35*x,5*y);
moveto(25*x,5*y); lineto(25*x,8*y);
moveto(35*x,5*y); lineto(35*x,8*y);
moveto(35*x,8*y); lineto(45*x,8*y);
moveto(45*x,8*y); lineto(35*x,11*y);
moveto(45*x,8*y); lineto(45*x,11*y);
moveto(45*x,8*y); lineto(45*x,5*y);
moveto(45*x,5*y); lineto(55*x,8*y);
moveto(55*x,8*y); lineto(55*x,5*y);
}
```

```
/***********************************************************************/
/* This routine gives the step by step solution to the exercise          */
/***********************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  /********************************************************************/
  outtextxy(62*x,5*y,"L");
  outtextxy(69*x,9*y/2,"EDGES");
  outtextxy(67*x,5*y,"TO CHECK");
  outtextxy(78*x,5*y," Wt.");
  outtextxy(86*x,5*y,"T");
  moveto(60*x,11*y/2); lineto(65*x,11*y/2);
  moveto(66*x,11*y/2); lineto(75*x,11*y/2);
  moveto(77*x,11*y/2); lineto(82*x,11*y/2);
  moveto(84*x,11*y/2); lineto(89*x,11*y/2);
  /********************************************************************/
  outtextxy(62*x,6*y,"A");
  Pause(30*x,24*y);
  outtextxy(69*x,6*y,"(A,B)");
  outtextxy(79*x,6*y,"4");
  outtextxy(69*x,7*y,"(A,E)");
  outtextxy(79*x,7*y,"3");
  Pause(30*x,24*y);
  outtextxy(84*x,7*y,"(A,E)");
  setcolor(backcolor);
  moveto(25*x,5*y); lineto(25*x,8*y);
  setlinestyle(3,0,3);
  setcolor(forecolor);
  moveto(25*x,5*y); lineto(25*x,8*y); /* add (A, E) to T */
  setlinestyle(0,0,3);
  moveto(69*x,7*y); lineto(74*x,7*y); /* delete (A,E) from the list*/
  Pause(30*x,24*y);
  setcolor(backcolor);
```

```
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(62*x,8*y,"E");
Pause(30*x,24*y);
outtextxy(69*x,8*y,"(E, F)");
outtextxy(79*x,8*y,"6");
Pause(30*x,24*y);
outtextxy(84*x,6*y,"(A,B)");
setcolor(backcolor);
moveto(25*x,5*y);  lineto(35*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,5*y);  lineto(35*x,5*y); /* add (A, B) to T */
setlinestyle(0,0,3);
moveto(69*x,6*y); lineto(74*x,6*y); /* delete (A,B) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(62*x,9*y,"B");
Pause(30*x,24*y);
outtextxy(69*x,9*y,"(B,F)");
outtextxy(79*x,9*y,"5");
Pause(30*x,24*y);
outtextxy(84*x,9*y,"(B,F)");
setcolor(backcolor);
moveto(35*x,5*y);  lineto(35*x,8*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,5*y);  lineto(35*x,8*y); /* add (B, F) to T */
setlinestyle(0,0,3);
moveto(69*x,9*y); lineto(74*x,9*y); /* delete (B, F) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
```

```
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(62*x,10*y,"F");
Pause(30*x,24*y);
outtextxy(69*x,10*y,"(F,G)");
outtextxy(79*x,10*y,"1");
outtextxy(69*x,11*y,"(F,I)");
outtextxy(79*x,11*y,"4");
Pause(30*x,24*y);
outtextxy(84*x,10*y,"(F,G)");
setcolor(backcolor);
moveto(35*x,8*y);  lineto(45*x,8*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,8*y);  lineto(45*x,8*y); /* add (F, G) to T */
setlinestyle(0,0,3);
moveto(69*x,10*y); lineto(74*x,10*y); /* delete (F, G) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(62*x,12*y,"G");
Pause(30*x,24*y);
outtextxy(69*x,12*y,"(G, C)");
outtextxy(79*x,12*y,"2");
outtextxy(69*x,13*y,"(G, H)");
outtextxy(79*x,13*y,"4");
outtextxy(69*x,14*y,"(G, I)");
outtextxy(79*x,14*y,"2");
outtextxy(69*x,15*y,"(G, J)");
outtextxy(79*x,15*y,"1");
Pause(30*x,24*y);
outtextxy(84*x,15*y,"(G,J)");
setcolor(backcolor);
```

```
moveto(45*x,8*y);  lineto(45*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,8*y);  lineto(45*x,11*y); /* add (G, J) to T */
setlinestyle(0,0,3);
moveto(69*x,15*y); lineto(74*x,15*y); /* delete (G, J) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***********************************************************************/
outtextxy(62*x,16*y,"J");
Pause(30*x,24*y);
outtextxy(69*x,16*y,"(J,I)");
outtextxy(79*x,16*y,"3");
Pause(30*x,24*y);
outtextxy(84*x,12*y,"(G,C)");
setcolor(backcolor);
moveto(45*x,8*y);  lineto(45*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,8*y);  lineto(45*x,5*y); /* add (G, C) to T */
setlinestyle(0,0,3);
moveto(69*x,12*y); lineto(74*x,12*y); /* delete (G, C) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***********************************************************************/
outtextxy(62*x,17*y,"C");
Pause(30*x,24*y);
outtextxy(69*x,17*y,"(C, D)");
outtextxy(79*x,17*y,"2");
outtextxy(69*x,18*y,"(C, H)");
outtextxy(79*x,18*y,"2");
Pause(30*x,24*y);
```

```
outtextxy(84*x,14*y,"(G,I)");
setcolor(backcolor);
moveto(45*x,8*y); lineto(35*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,8*y); lineto(35*x,11*y); /* add (G, I) to T */
setlinestyle(0,0,3);
moveto(69*x,14*y); lineto(74*x,14*y); /* delete (G, I) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(62*x,19*y,"I");
Pause(30*x,24*y);
outtextxy(84*x,18*y,"(C,H)");
setcolor(backcolor);
moveto(45*x,5*y); lineto(55*x,8*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,5*y); lineto(55*x,8*y); /* add (C, H) to T */
setlinestyle(0,0,3);
moveto(69*x,18*y); lineto(74*x,18*y); /* delete (C, H) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(62*x,20*y,"H");
Pause(30*x,24*y);
outtextxy(69*x,20*y,"(H, D)");
outtextxy(79*x,20*y,"2");
Pause(30*x,24*y);
outtextxy(84*x,20*y,"(H,D)");
setcolor(backcolor);
moveto(55*x,5*y); lineto(55*x,8*y);
```

```
    setlinestyle(3,0,3);
    setcolor(forecolor);
    moveto(55*x,5*y);  lineto(55*x,8*y); /* add (H, D) to T */
    setlinestyle(0,0,3);
    moveto(69*x,20*y); lineto(74*x,20*y); /* delete (H, D) from the list*/
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /*********************************************************************/
    Pause(30*x,24*y);
    setcolor(backcolor);          /* Clean the game field  again */
    bar(2*x,13*y,179*x/2,49*y/2);
    bar(59*x,7*y/2,179*x/2,49*y/2);
    setcolor(forecolor);
    moveto(25*x,5*y); lineto(35*x,5*y);
    moveto(25*x,5*y); lineto(25*x,8*y);
    moveto(35*x,5*y); lineto(35*x,8*y);
    moveto(35*x,8*y); lineto(45*x,8*y);
    moveto(45*x,8*y); lineto(35*x,11*y);
    moveto(45*x,8*y); lineto(45*x,11*y);
    moveto(45*x,8*y); lineto(45*x,5*y);
    moveto(45*x,5*y); lineto(55*x,8*y);
    moveto(55*x,8*y); lineto(55*x,5*y);
  }
```

```
/************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
    }
    switch (ch)          {
     case 'y': in_the_exercise = 0;
            break;
     case 'Y': in_the_exercise = 0;
            break;
     case 'n': setcolor(backcolor);
            bar(46*x,37*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;
     case 'N': setcolor(backcolor);
            bar(46*x,37*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;
     default : break;
     }
}
```

```
/* PROGRAM  : q442.c
   AUTHOR     : Atilla BAKAN
   DATE       : Mar. 22, 1990
   REVISED    : Apr. 22, 1990

   DESCRIPTION : This program contains the second exercise about the minimal
                 spanning trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)     _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)    _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/* Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example          (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
 int in_the_exercise = 1;


/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/***********************************************************************/
/* This function is used for including drivers to the executable code     */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/***********************************************************************/
/* This fuction initializes the necessary graphical routines               */
/***********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*********************************************************************/
  settext();
  /*********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
       }
  forecolor = WHITE;
 }


/**********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(32*x,24*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(31*x,23*y,69*x,97*y/4);
       setcolor(quitcolor);
    }
    switch (ch)         {
```

```
    case 'y': closegraph();
            videoinit();
            exit(0);
            break;

    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;

    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;

    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;

    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
```

```
/**********************************************************************/
/* This function sets the text default values                          */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}


/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }


/**********************************************************************/
/* main routine which calls exer routine                               */
/**********************************************************************/
void main()
{
  exer();
}
```

```
/**********************************************************************/
/* Routine that asks the question, then depending on the user's answer        */
/* makes necessary explanations                                                          */
/**********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  2");
   /*******************************************************************/
   outtextxy(2*x,2*y,"Use Kruskal's algorithm to find a minimal spanning tree. (Start
                        at A. If ");
   outtextxy(2*x,3*y,"there is a choice of edges select edges according to alphabeti-
                        cal order.)");
   /*******************************************************************/
   pieslice(25*x,5*y,0,359,2);     /* A */
   pieslice(35*x,5*y,0,359,2);     /* B */
   pieslice(45*x,5*y,0,359,2);     /* C */
   pieslice(55*x,5*y,0,359,2);     /* D */
   pieslice(25*x,8*y,0,359,2);     /* E */
   pieslice(35*x,8*y,0,359,2);     /* F */
   pieslice(45*x,8*y,0,359,2);     /* G */
   pieslice(55*x,8*y,0,359,2);     /* H */
   pieslice(35*x,11*y,0,359,2);     /* I */
   pieslice(45*x,11*y,0,359,2);     /* J */
  /********************************************************************/
   outtextxy(25*x,9*y/2,"A");
   outtextxy(35*x,9*y/2,"B");
   outtextxy(45*x,9*y/2,"C");
   outtextxy(55*x,9*y/2,"D");
   outtextxy(23*x,8*y,"E");
   outtextxy(33*x,17*y/2,"F");
outtextxy(47*x,17*y/2,"G");
outtextxy(57*x,8*y,"H");
outtextxy(35*x,23*y/2,"I");
outtextxy(45*x,23*y/2,"J");
/**********************************************************************/
outtextxy(30*x,9*y/2,"4");          /* (A, B) */
outtextxy(50*x,9*y/2,"3");          /* (C, D) */
outtextxy(23*x,13*y/2,"3");         /* (A, E) */
outtextxy(36*x,13*y/2,"5");         /* (B, F) */
outtextxy(43*x,13*y/2,"2");         /* (C, G) */
outtextxy(56*x,13*y/2,"2");         /* (D, H) */
outtextxy(50*x,6*y,"2");            /* (C, H) */
outtextxy(30*x,15*y/2,"6");         /* (E, F) */
outtextxy(40*x,15*y/2,"1");         /* (F, G) */
outtextxy(49*x,15*y/2,"4");         /* (G, H) */
outtextxy(33*x,19*y/2,"4");         /* (F, I) */
outtextxy(40*x,23*y/2,"3");         /* (I, J) */
outtextxy(46*x,19*y/2,"1");         /* (G, J) */
outtextxy(38*x,19*y/2,"2");         /* (G, I) */
/**********************************************************************/
moveto(25*x,5*y); lineto(35*x,5*y); lineto(35*x,8*y);
lineto(25*x,8*y); lineto(25*x,5*y);
moveto(45*x,8*y); lineto(45*x,11*y);lineto(35*x,11*y);
lineto(35*x,8*y); lineto(45*x,8*y);
moveto(35*x,11*y);lineto(45*x,8*y);
moveto(45*x,8*y); lineto(45*x,5*y); lineto(55*x,5*y);
lineto(55*x,8*y);lineto(45*x,8*y);
moveto(45*x,5*y);lineto(55*x,8*y);
/**********************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"     a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"     b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"     c) I want to see step by step solution.");
outtextxy(15*x,18*y,"     d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
```

```
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
/************************************************************************/
 while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,19*y,"    Please type a, b, c or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
   setcolor(backcolor);
   bar(50*x,37*y/2,88*x,20*y);
   setcolor(forecolor);
   }
  }
   switch (Ch)         {
   case 'a': outtextxy(47*x,19*y,"a");
     outtextxy(52*x,19*y,"You want to see the algorithm ");
     outtextxy(52*x,20*y,"again. Press any key to continue.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,37*y/2,179*x/2,21*y);
     bar(2*x,13*y,179*x/2,49*y/2);
     setcolor(forecolor);
     show_alg();
     break;
  case 'b': outtextxy(47*x,19*y,"b");
     outtextxy(52*x,19*y,"You want to compare your solu-");
     outtextxy(52*x,20*y,"tion with ours. So press any  ");
     outtextxy(52*x,21*y,"key to see it.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x.37*y/2,179*x/2,22*y);
     bar(2*x,13*y,179*x/2,49*y/2);
     setcolor(forecolor);
     compare_solutions();
     break;
  case 'c': outtextxy(47*x,19*y,"c");
```

```
        outtextxy(52*x,19*y,"You want to see step by step");
        outtextxy(52*x,20*y,"solution. So press any key to ");
        outtextxy(52*x,21*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
    }
  }
  closegraph();
}



/***********************************************************************/
/* This routine gives Prim's minimal spanning tree algorithm           */
/***********************************************************************/
static void show_alg(void)
{
  outtextxy(5*x,15*y,"Step 1 . Order the edges from smallest weight to largest.");
  outtextxy(5*x,17*y,"Step 2 . Add the edges in order, as long as a cycle is not");
  outtextxy(5*x,18*y,"        created. T can be disconnected until it's completed." );
    outtextxy(5*x,20*y,"Step 3 . If all nodes are visited STOP, or else GO TO Step
2.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,13*y,88*x,49*y/2);
  setcolor(forecolor);
}
```

```
/***************************************************************************/
/* This routine gives the solution to the exercise to be compared.         */
/***************************************************************************/
static void compare_solutions(void)
{

  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,13*y,179*x/2,49*y/2);
  /*************************************************************************/
  moveto(25*x,5*y); lineto(35*x,5*y);
  moveto(25*x,5*y); lineto(25*x,8*y);
  moveto(35*x,5*y); lineto(35*x,8*y);
  moveto(35*x,8*y); lineto(45*x,8*y);
  moveto(45*x,8*y); lineto(35*x,11*y);
  moveto(45*x,8*y); lineto(45*x,11*y);
  moveto(45*x,8*y); lineto(45*x,5*y);
  moveto(45*x,5*y); lineto(55*x,5*y);
  moveto(55*x,8*y); lineto(55*x,5*y);
  /*************************************************************************/
  setcolor(forecolor);
  setlinestyle(3,0,3);
  /*************************************************************************/
  moveto(25*x,5*y); lineto(35*x,5*y);
  moveto(25*x,5*y); lineto(25*x,8*y);
  moveto(35*x,5*y); lineto(35*x,8*y);
  moveto(35*x,8*y); lineto(45*x,8*y);
  moveto(45*x,8*y); lineto(35*x,11*y);
  moveto(45*x,8*y); lineto(45*x,11*y);
  moveto(45*x,8*y); lineto(45*x,5*y);
  moveto(45*x,5*y); lineto(55*x,5*y);
  moveto(55*x,8*y); lineto(55*x,5*y);
  /*************************************************************************/
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
```

```
   setcolor(forecolor);
   /*****************************************************************/
   moveto(25*x,5*y); lineto(35*x,5*y);
   moveto(25*x,5*y); lineto(25*x,8*y);
   moveto(35*x,5*y); lineto(35*x,8*y);
   moveto(35*x,8*y); lineto(45*x,8*y);
   moveto(45*x,8*y); lineto(35*x,11*y);
   moveto(45*x,8*y); lineto(45*x,11*y);
   moveto(45*x,8*y); lineto(45*x,5*y);
   moveto(45*x,5*y); lineto(55*x,5*y);
   moveto(55*x,8*y); lineto(55*x,5*y);
}

/******************************************************************/
/* This routine gives the step by step solution to the exercise          */
/******************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   outtextxy(61*x,5*y,"EDGES TO CHECK");
   outtextxy(78*x,5*y,"WEIGHT");
   moveto(60*x,11*y/2); lineto(75*x,11*y/2);
   moveto(77*x,11*y/2); linetc(85*x,11*y/2);
   /*****************************************************************/
   outtextxy(65*x,6*y,"(F,G)");  outtextxy(81*x,6*y,"1");
   outtextxy(65*x,7*y,"(G,J)");  outtextxy(81*x,7*y,"1");
   outtextxy(65*x,8*y,"(C,H)");  outtextxy(81*x,10*y,"2");
   outtextxy(65*x,9*y,"(D,H)");  outtextxy(81*x,11*y,"2");
   outtextxy(65*x,10*y,"(G,C)"); outtextxy(81*x,9*y,"2");
   outtextxy(65*x,11*y,"(G,I)"); outtextxy(81*x,8*y,"2");
   outtextxy(65*x,12*y,"(A,E)"); outtextxy(81*x,12*y,"3");
   outtextxy(65*x,13*y,"(C,D)"); outtextxy(81*x,13*y,"3");
   outtextxy(65*x,14*y,"(I,J)"); outtextxy(81*x,14*y,"3");
outtextxy(65*x,15*y,"(A,B)"); outtextxy(81*x,15*y,"4");
outtextxy(65*x,16*y,"(F,I)"); outtextxy(81*x,16*y,"4");
outtextxy(65*x,17*y,"(G,H)"); outtextxy(81*x,17*y,"4");
outtextxy(65*x,18*y,"(B,F)"); outtextxy(81*x,18*y,"5");
outtextxy(65*x,19*y,"(E,F)"); outtextxy(81*x,19*y,"6");
/*******************************************************************/
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(25*x,5*y); lineto(35*x,5*y); lineto(35*x,8*y);
lineto(25*x,8*y); lineto(25*x,5*y);
moveto(45*x,8*y); lineto(45*x,11*y);lineto(35*x,11*y);
lineto(35*x,8*y); lineto(45*x,8*y);
moveto(35*x,11*y);lineto(45*x,8*y);
moveto(45*x,8*y); lineto(45*x,5*y); lineto(55*x,5*y);
lineto(55*x,8*y);lineto(45*x,8*y);
moveto(45*x,5*y);lineto(55*x,8*y);
setcolor(forecolor);
/*******************************************************************/
pieslice(25*x,5*y,0,359,2);    /* A */
pieslice(35*x,5*y,0,359,2);    /* B */
pieslice(45*x,5*y,0,359,2);    /* C */
pieslice(55*x,5*y,0,359,2);    /* D */
pieslice(25*x,8*y,0,359,2);    /* E */
pieslice(35*x,8*y,0.359,2);    /* F */
pieslice(45*x,8*y,0,359,2);    /* G */
pieslice(55*x,8*y,0,359,2);    /* H */
pieslice(35*x,11*y,0,359,2);    /* I */
pieslice(45*x,11*y,0,359,2);    /* J */
Pause(30*x,24*y);
/*******************************************************************/
moveto(65*x,6*y); lineto(70*x,6*y);    /* (F,G) */
moveto(35*x.8*y); lineto(45*x,8*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y.50*x,49*y/2);
setcolor(forecolor);
```

```
/*******************************************************************/
moveto(65*x,7*y); lineto(70*x,7*y);    /* (G,J) */
moveto(45*x,8*y); lineto(45*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(65*x,8*y); lineto(70*x,8*y);    /* (D,H) */
moveto(55*x,5*y); lineto(55*x,8*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(65*x,9*y); lineto(70*x,9*y);    /* (G,C) */
moveto(45*x,5*y); lineto(45*x,8*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(65*x,10*y); lineto(70*x,10*y);    /* (G,I) */
moveto(45*x,8*y); lineto(35*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(65*x,11*y); lineto(70*x,11*y);    /* (A,E) */
moveto(25*x,5*y); lineto(25*x,8*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
```

```
moveto(65*x,12*y); lineto(70*x,12*y);   /* (C,D) */
moveto(45*x,5*y);  lineto(55*x,5*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
moveto(65*x,13*y); lineto(81*x,13*y);   /* (I,J) */
moveto(65*x,14*y); lineto(70*x,14*y);   /* (A,B) */
moveto(25*x,5*y);  lineto(35*x,5*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
moveto(65*x,15*y); lineto(81*x,15*y);   /* (F,I) */
moveto(65*x,16*y); lineto(81*x,16*y);   /* (G,H) */
moveto(65*x,17*y); lineto(70*x,17*y);   /* (B,F) */
moveto(35*x,5*y);  lineto(35*x,8*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
outtextxy(65*x,21*y,"We are done.");
/*********************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);          /* Clean the game field  again */
bar(2*x,13*y,179*x/2,49*y/2);
bar(59*x,7*y/2,179*x/2,49*y/2);
setcolor(forecolor);
moveto(25*x,8*y); lineto(35*x,8*y);
moveto(35*x,8*y); lineto(35*x,11*y);
moveto(35*x,11*y);lineto(45*x,11*y);
moveto(45*x,8*y); lineto(55*x,8*y);
moveto(45*x,5*y); lineto(55*x,8*y);
}
```

```
/***********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
   switch (ch)        {
    case 'y': in_the_exercise = 0;
          break;
    case 'Y': in_the_exercise = 0;
          break;

    case 'n': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    case 'N': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    default : break;
    }
}
```

```
/* PROGRAM   : q443.c
   AUTHOR    : Atilla BAKAN
   DATE      : Mar. 22, 1990
   REVISED   : Apr. 22, 1990

   DESCRIPTION : This program contains the third exercise about the minimal
                 spanning trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk          find_t
  #define ff_name         name
#elif defined(__ZTC__)                   /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name          name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;


/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/**********************************************************************/
/* This function is used for including drivers to the executable code   */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/**********************************************************************/
/* This fuction initializes the necessary graphical routines           */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**********************************************************************/
  settext();
  /**********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
  else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
}


/**********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(32*x,24*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(31*x,23*y,69*x,97*y/4);
       setcolor(quitcolor);
    }
   switch (ch)          {
    case 'y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}


/************************************************************************/
/* This function sets the text default values                          */
/************************************************************************/
static void settext(void)
{
 settextstyle(0,0,0);
 setlinestyle(0,4,3);
 settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/***********************************************************************/
/* main routine that calls exer routine                                */
/***********************************************************************/
void main()
{
  exer();
}


/***********************************************************************/
/* Routine that asks the question, then depending on the user's answer */
/* makes necessary explanations                                        */
/***********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  3");
   /*********************************************************************/
   outtextxy(2*x,2*y,"Use Prim's algorithm to find a minimal spanning tree. (Start at
                    A. If there");
outtextxy(2*x,3*y,"is a choice of edges select edges according to alphabetical
                    order.)");
/**********************************************************************/
pieslice(25*x,5*y,0,359,2);       /* A */
pieslice(25*x,11*y,0,359,2);      /* B */
pieslice(55*x,5*y,0,359,2);       /* C */
pieslice(55*x,11*y,0,359,2);      /* D */
pieslice(35*x,7*y,0,359,2);       /* E */
pieslice(45*x,7*y,0,359,2);       /* F */
pieslice(35*x,9*y,0,359,2);       /* G */
pieslice(45*x,9*y,0,359,2);       /* H */
/**********************************************************************/
outtextxy(25*x,9*y/2,"A");
outtextxy(25*x,23*y/2,"B");
outtextxy(55*x,9*y/2,"C");
outtextxy(55*x,23*y/2,"D");
outtextxy(33*x,7*y,"E");
outtextxy(46*x,7*y,"F");
outtextxy(33*x,9*y,"G");
outtextxy(46*x,9*y,"H");
/**********************************************************************/
outtextxy(40*x,9*y/2,"4");        /* (A, C) */
outtextxy(23*x,8*y,"6");          /* (A, B) */
outtextxy(32*x,6*y,"2");          /* (A, E) */
outtextxy(56*x,8*y,"5");          /* (C, D) */
outtextxy(40*x,23*y/2,"3");        /* (B, D) */
outtextxy(33*x,8*y,"2");          /* (E, G) */
outtextxy(40*x,13*y/2,"2");        /* (E, F) */
outtextxy(42*x,8*y,"2");          /* (F, G) */
outtextxy(46*x,8*y,"1");          /* (F, H) */
outtextxy(40*x,19*y/2,"3");        /* (G, H) */
outtextxy(52*x,10*y,"1");          /* (H, D) */
/**********************************************************************/
moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
lineto(45*x,7*y);  lineto(35*x,7*y); lineto(35*x,9*y);
lineto(45*x,9*y);
moveto(45*x,7*y);lineto(35*x,9*y);
moveto(25*x,5*y);lineto(35*x,7*y);
/*********************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
/*********************************************************************/
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,'    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
   }
    switch (Ch)        {
    case 'a': outtextxy(47*x,19*y,"a");
      outtextxy(52*x,19*y,"You want to see the algorithm ");
      outtextxy(52*x,20*y,"again. Press any key to continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,21*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      show_alg();
      break;
   case 'b': outtextxy(47*x,19*y,"b");
```

```
        outtextxy(52*x,19*y,"You want to compare your solu-");
        outtextxy(52*x,20*y,"tion with ours. So press any  ");
        outtextxy(52*x,21*y,"key to see it.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        compare_solutions();
        break;
      case 'c': outtextxy(47*x,19*y,"c");
        outtextxy(52*x,19*y,"You want to see step by step");
        outtextxy(52*x,20*y,"solution. So press any key to ");
        outtextxy(52*x,21*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
     }
   }
   closegraph();
 }
```

```
/*************************************************************************/
/* This routine gives Prim's minimal spanning tree algorithm              */
/*************************************************************************/
static void show_alg(void)
{
  outtextxy(5*x,15*y,"Step 1 . Pick an arbitrary initial vertex x.");
  outtextxy(5*x,16*y,"          L = { x }, T = 0");
  outtextxy(5*x,17*y,"Step 2 . If |L| = n then stop and output T.");
   outtextxy(5*x,18*y,"Step 3 . Else, find all edges with one vertex Ui in L and the
other Vj ");
   outtextxy(5*x,19*y,"          which is not in L yet. Pick the one with least weight, (U,
V)");
  outtextxy(5*x,20*y,"          L <- L U {V}");
  outtextxy(5*x,21*y,"          T <- T U {U, V}");
  outtextxy(5*x,22*y,"          go to Step 2.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,13*y,88*x,49*y/2);
  setcolor(forecolor);
}
```

```
/**********************************************************************/
/* This routine gives the solution to the exercise to be compared.   */
/**********************************************************************/
static void compare_solutions(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,13*y,179*x/2,49*y/2);
  /*******************************************************************/
  moveto(55*x,5*y); lineto(25*x,5*y); lineto(35*x,7*y);
  lineto(45*x,7*y); lineto(45*x,9*y); lineto(55*x,11*y);
  lineto(25*x,11*y);
  moveto(35*x,7*y);lineto(35*x,9*y);
  /*******************************************************************/
  setcolor(forecolor);
  setlinestyle(3,0,3);
  /*******************************************************************/
  moveto(55*x,5*y); lineto(25*x,5*y); lineto(35*x,7*y);
  lineto(45*x,7*y); lineto(45*x,9*y); lineto(55*x,11*y);
  lineto(25*x,11*y);
  moveto(35*x,7*y);lineto(35*x,9*y);
  /*******************************************************************/
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  moveto(55*x,5*y); lineto(25*x,5*y); lineto(35*x,7*y);
  lineto(45*x,7*y); lineto(45*x,9*y); lineto(55*x,11*y);
  lineto(25*x,11*y);
  moveto(35*x,7*y);lineto(35*x,9*y);
}
```

```
/*************************************************************************/
/* This routine gives the step by step solution to the exercise         */
/*************************************************************************/
static void step_solution(void)
{

   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   /*********************************************************************/
   outtextxy(62*x,5*y,"L");
   outtextxy(69*x,9*y/2,"EDGES");
   outtextxy(67*x,5*y,"TO CHECK");
   outtextxy(78*x,5*y," Wt.");
   outtextxy(86*x,5*y,"T");
   moveto(60*x,11*y/2); lineto(65*x,11*y/2);
   moveto(66*x,11*y/2); lineto(75*x,11*y/2);
   moveto(77*x,11*y/2); lineto(82*x,11*y/2);
   moveto(84*x,11*y/2); lineto(89*x,11*y/2);
   /*********************************************************************/
   outtextxy(62*x,6*y,"A");
   Pause(30*x,24*y);
   outtextxy(69*x,6*y,"(A,B)");
   outtextxy(79*x,6*y,"6");
   outtextxy(69*x,7*y,"(A,C)");
   outtextxy(79*x,7*y,"4");
   outtextxy(69*x,8*y,"(A,E)");
   outtextxy(79*x,8*y,"2");
   Pause(30*x,24*y);
   outtextxy(84*x,8*y,"(A,E)");
   setcolor(backcolor);
   moveto(25*x,5*y); lineto(35*x,7*y);
   setlinestyle(3,0,3);
   setcolor(forecolor);
   moveto(25*x,5*y); lineto(35*x,7*y); /* add (A, E) to T */
   setlinestyle(0,0,3);
```

```
moveto(69*x,8*y); lineto(74*x,8*y); /* delete (A,E) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(62*x,9*y,"E");
Pause(30*x,24*y);
outtextxy(69*x,9*y,"(E,F)");
outtextxy(79*x,9*y,"2");
outtextxy(69*x,10*y,"(E,G)");
outtextxy(79*x,10*y,"2");
Pause(30*x,24*y);
outtextxy(84*x,6*y,"(E,F)");
setcolor(backcolor);
moveto(35*x,7*y);  lineto(45*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y);  lineto(45*x,7*y); /* add (E, F) to T */
setlinestyle(0,0,3);
moveto(69*x,9*y); lineto(74*x,9*y); /* delete (A,B) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(62*x,11*y,"F");
Pause(30*x,24*y);
outtextxy(69*x,11*y,"(F,G)");
outtextxy(79*x,11*y,"2");
outtextxy(69*x,12*y,"(F,H)");
outtextxy(79*x,12*y,"1");
Pause(30*x,24*y);
outtextxy(84*x,9*y,"(F,H)");
setcolor(backcolor);
moveto(45*x,7*y);  lineto(45*x,9*y);
```

```
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,7*y);  lineto(45*x,9*y); /* add (F, H) to T */
setlinestyle(0,0,3);
moveto(69*x,12*y); lineto(74*x,12*y); /* delete (F, H) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(62*x,13*y,"H");
Pause(30*x,24*y);
outtextxy(69*x,13*y,"(H,D)");
outtextxy(79*x,13*y,"1");
outtextxy(69*x,14*y,"(H,G)");
outtextxy(79*x,14*y,"3");
Pause(30*x,24*y);
outtextxy(84*x,13*y,"(H,D)");
setcolor(backcolor);
moveto(45*x,9*y);  lineto(55*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,9*y);  lineto(55*x,11*y); /* add (H, D) to T */
setlinestyle(0,0,3);
moveto(69*x,13*y); lineto(74*x,13*y); /* delete (H, D) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(62*x,15*y,"D");
Pause(30*x,24*y);
outtextxy(69*x,15*y,"(D,B)");
outtextxy(79*x,15*y,"3");
outtextxy(69*x,16*y,"(D,C)");
outtextxy(79*x,16*y,"5");
```

```
Pause(30*x,24*y);
outtextxy(84*x,10*y,"(E,G)");
setcolor(backcolor);
moveto(35*x,7*y); lineto(35*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y); lineto(35*x,9*y); /* add (E, G) to T */
setlinestyle(0,0,3);
moveto(69*x,10*y); lineto(74*x,10*y); /* delete (E, G) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(62*x,17*y,"G");
Pause(30*x,24*y);
outtextxy(69*x,17*y,"-");
outtextxy(79*x,17*y,"-");
Pause(30*x,24*y);
outtextxy(84*x,15*y,"(D,B)");
setcolor(backcolor);
moveto(25*x,11*y); lineto(55*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,11*y); lineto(55*x,11*y); /* add (D, B) to T */
setlinestyle(0,0,3);
moveto(69*x,15*y); lineto(74*x,15*y); /* delete (D, B) from the list*/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(62*x,18*y,"B");
Pause(30*x,24*y);
outtextxy(69*x,18*y,"-");
outtextxy(79*x,18*y,"-");
```

```
  outtextxy(84*x,7*y,"(A,C)");
  setcolor(backcolor);
  moveto(25*x,5*y); lineto(55*x,5*y);
  setlinestyle(3,0,3);
  setcolor(forecolor);
  moveto(25*x,5*y); lineto(55*x,5*y); /* add (A, C) to T */
  setlinestyle(0,0,3);
  moveto(69*x,7*y); lineto(74*x,7*y); /* delete (A, C) from the list*/
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  outtextxy(62*x,20*y,"We are done.");
  /***********************************************************************/
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field  again */
  bar(2*x,13*y,179*x/2,49*y/2);
  bar(59*x,7*y/2,179*x/2,49*y/2);
  setcolor(forecolor);
  moveto(55*x,5*y); lineto(25*x,5*y); lineto(35*x,7*y);
  lineto(45*x,7*y); lineto(45*x,9*y); lineto(55*x,11*y);
  lineto(25*x,11*y);
  moveto(35*x,7*y);lineto(35*x,9*y);
}
```

```
/*****************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n -- ");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,23*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,22*y,179*x/2,49*y/2);
       setcolor(forecolor);
   }
    switch (ch)          {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;

    case 'n': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2.22*y);
           setcolor(forecolor);
           break;

    case 'N': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    default : break;
    }
}
```

```
/* PROGRAM   : q444.c
   AUTHOR    : Atilla BAKAN
   DATE      : Mar. 22, 1990
   REVISED   : Apr. 22, 1990

   DESCRIPTION : This program contains the forth exercise about the minimal
                 spanning trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                  /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
   #define bioskey(a)        _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)       _dos_findnext(a)
   #define ffblk             find_t
   #define ff_name           name
#elif defined(__ZTC__)                   /* Zortech C/C++ */
   #define ffblk             FIND
   #define ff_name           name
   #define ff_attrib         attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example          (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY,
/************************************************************************/
/* This function is used for including drivers to the executable code       */
/************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/************************************************************************/
/* This fuction initializes the necessary graphical routines               */
/************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**********************************************************************/
  settext();
  /**********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }



/***********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   default : break;
   }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);    /* restore any hidden hot keys */
}


/*************************************************************************/
/* This function sets the text default values                           */
/*************************************************************************/
static void settext(void)
{
 settextstyle(0,0,0);
 setlinestyle(0,4,3);
 settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/************************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }


/************************************************************************/
/* main routine  which calls exer routine                               */
/************************************************************************/
void main()
{
  exer();
}
```

```
/**********************************************************************/
/* Routine that asks the question, then depending on the user's answer        */
/* makes necessary explanations                                                   */
/**********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  4");
   /*******************************************************************/
   outtextxy(2*x,2*y,"Use Kruskal's algorithm to find a minimal spanning tree. (Start
                              at A. If ");
   outtextxy(2*x,3*y,"there is a choice of edges select edges according to alphabeti-
                              cal order.)");
   /********************************************************************/
   pieslice(25*x,5*y,0,359,2);      /* A */
   pieslice(25*x,11*y,0,359,2);     /* B */
   pieslice(55*x,5*y,0,359,2);      /* C */
   pieslice(55*x,11*y,0,359,2);     /* D */
   pieslice(35*x,7*y,0,359,2);      /* E */
   pieslice(45*x,7*y,0,359,2);      /* F */
   pieslice(35*x,9*y,0,359,2);      /* G */
   pieslice(45*x,9*y,0,359,2);      /* H */
   /*********-**********************************************************/
   outtextxy(25*x,9*y/2,"A");
   outtextxy(25*x,23*y/2,"B");
   outtextxy(55*x,9*y/2,"C");
   outtextxy(55*x,23*y/2,"D");
   outtextxy(33*x,7*y,"E");
   outtextxy(46*x,7*y,"F");
   outtextxy(33*x,9*y,"G");
   outtextxy(46*x,9*y,"H");
/**********************************************************************/
outtextxy(40*x,9*y/2,"4");        /* (A, C) */
outtextxy(23*x,8*y,"6");          /* (A, B) */
outtextxy(32*x,6*y,"2");          /* (A, E) */
outtextxy(56*x,8*y,"5");          /* (C, D) */
outtextxy(40*x,23*y/2,"3");       /* (B, D) */
outtextxy(33*x,8*y,"2");          /* (E, G) */
outtextxy(40*x,13*y/2,"2");       /* (E, F) */
outtextxy(42*x,8*y,"2");          /* (F, G) */
outtextxy(46*x,8*y,"1");          /* (F, H) */
outtextxy(40*x,19*y/2,"3");       /* (G, H) */
outtextxy(52*x,10*y,"1");         /* (H, D) */
/**********************************************************************/
moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
lineto(45*x,7*y);  lineto(35*x,7*y); lineto(35*x,9*y);
lineto(45*x,9*y);
moveto(45*x,7*y);lineto(35*x,9*y);
moveto(25*x,5*y);lineto(35*x,7*y);
/**********************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
/**********************************************************************/
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
```

```
 bar(50*x,37*y/2,88*x,20*y);
 setcolor(forecolor);
 }
}
 switch (Ch)         {
 case 'a': outtextxy(47*x,19*y,"a");
   outtextxy(52*x,19*y,"You want to see the algorithm ");
   outtextxy(52*x,20*y,"again. Press any key to continue.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(50*x,37*y/2,179*x/2,21*y);
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   show_alg();
   break;
case 'b': outtextxy(47*x,19*y,"b");
   outtextxy(52*x,19*y,"You want to compare your solu-");
   outtextxy(52*x,20*y,"tion with ours. So press any  ");
   outtextxy(52*x,21*y,"key to see it.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(50*x,37*y/2,179*x/2,22*y);
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   compare_solutions();
   break;
case 'c': outtextxy(47*x,19*y,"c");
   outtextxy(52*x,19*y,"You want to see step by step");
   outtextxy(52*x,20*y,"solution. So press any key to ");
   outtextxy(52*x,21*y,"continue.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(50*x,37*y/2,179*x/2,22*y);
   bar(2*x,13*y,179*x/2,49*y/2);
   setcolor(forecolor);
   step_solution();
```

```
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
    }
  }
  closegraph();
}


/***********************************************************************/
/* This routine gives Prim's minimal spanning tree algorithm              */
/***********************************************************************/
static void show_alg(void)
{
  outtextxy(5*x,15*y,"Step 1 . Order the edges from smallest weight to largest.");
  outtextxy(5*x,17*y,"Step 2 . Add the edges in order, as long as a cycle is not");
  outtextxy(5*x,18*y,"       created. T can be disconnected until it's completed." );
    outtextxy(5*x,20*y,"Step 3 . If all nodes are visited STOP, or else GO TO Step
2.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,13*y,88*x,49*y/2);
  setcolor(forecolor);
}
```

```
/*************************************************************************/
/* This routine gives the solution to the exercise to be compared.      */
/*************************************************************************/
static void compare_solutions(void)
{

   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,13*y,179*x/2,49*y/2);
   /*********************************************************************/
   moveto(55*x,5*y); lineto(25*x,5*y); lineto(35*x,7*y);
   lineto(45*x,7*y); lineto(45*x,9*y); lineto(55*x,11*y);
   lineto(25*x,11*y);
   moveto(35*x,7*y);lineto(35*x,9*y);
   /*********************************************************************/
   setcolor(forecolor);
   setlinestyle(3,0,3);
   /*********************************************************************/
   moveto(55*x,5*y); lineto(25*x,5*y); lineto(35*x,7*y);
   lineto(45*x,7*y); lineto(45*x,9*y); lineto(55*x,11*y);
   lineto(25*x,11*y);
   moveto(35*x,7*y);lineto(35*x,9*y);
   /*********************************************************************/
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,70*x,49*y/2);
   setcolor(forecolor);
   /*********************************************************************/
   moveto(55*x,5*y); lineto(25*x,5*y); lineto(35*x,7*y);
   lineto(45*x,7*y); lineto(45*x,9*y); lineto(55*x,11*y);
   lineto(25*x,11*y);
   moveto(35*x,7*y);lineto(35*x,9*y);
}
```

```
/**********************************************************************/
/* This routine gives the step by step solution to the exercise       */
/**********************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  outtextxy(61*x,5*y,"EDGES TO CHECK");
  outtextxy(78*x,5*y,"WEIGHT");
  moveto(60*x,11*y/2); lineto(75*x,11*y/2);
  moveto(77*x,11*y/2); lineto(85*x,11*y/2);
  /*******************************************************************/
  outtextxy(65*x,6*y,"(F,H)"); outtextxy(81*x,6*y,"1");
  outtextxy(65*x,7*y,"(H,D)"); outtextxy(81*x,7*y,"1");
  outtextxy(65*x,8*y,"(A,E)"); outtextxy(81*x,10*y,"2");
  outtextxy(65*x,9*y,"(E,F)"); outtextxy(81*x,11*y,"2");
  outtextxy(65*x,10*y,"(E,G)"); outtextxy(81*x,9*y,"2");
  outtextxy(65*x,11*y,"(F,G)"); outtextxy(81*x,8*y,"2");
  outtextxy(65*x,12*y,"(B,D)"); outtextxy(81*x,12*y,"3");
  outtextxy(65*x,13*y,"(G,H)"); outtextxy(81*x,13*y,"3");
  outtextxy(65*x,14*y,"(A,C)"); outtextxy(81*x,14*y,"3");
  outtextxy(65*x,15*y,"(C,D)"); outtextxy(81*x,15*y,"4");
  outtextxy(65*x,16*y,"(A,B)"); outtextxy(81*x,16*y,"4");
  /*******************************************************************/
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  moveto(55*x,11*y); lineto(55*x,5*y); lineto(25*x,5*y);
  lineto(25*x,11*y); lineto(55*x,11*y);lineto(45*x,9*y);
  lineto(45*x,7*y); lineto(35*x,7*y); lineto(35*x,9*y);
  lineto(45*x,9*y);
  moveto(45*x,7*y);lineto(35*x,9*y);
  moveto(25*x,5*y);lineto(35*x,7*y);
  setcolor(forecolor);
  /*******************************************************************/
```

```
pieslice(25*x,5*y,0,359,2);     /* A */
pieslice(25*x,11*y,0,359,2);    /* B */
pieslice(55*x,5*y,0,359,2);     /* C */
pieslice(55*x,11*y,0,359,2);    /* D */
pieslice(35*x,7*y,0,359,2);     /* E */
pieslice(45*x,7*y,0,359,2);     /* F */
pieslice(35*x,9*y,0,359,2);     /* G */
pieslice(45*x,9*y,0,359,2);     /* H */
Pause(30*x,24*y);
/*******************************************************************/
moveto(65*x,6*y); lineto(70*x,6*y);    /* (F,H) */
moveto(45*x,7*y); lineto(45*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(65*x,7*y); lineto(70*x,7*y);    /* (H,D) */
moveto(45*x,9*y); lineto(55*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(65*x,8*y); lineto(70*x,8*y);    /* (A,E) */
moveto(25*x,5*y); lineto(35*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
moveto(65*x,9*y); lineto(70*x,9*y);    /* (E,F) */
moveto(35*x,7*y); lineto(45*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
```

```
setcolor(forecolor);
/***********************************************************************/
moveto(65*x,10*y); lineto(70*x,10*y);   /* (E,G) */
moveto(35*x,7*y);  lineto(35*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***********************************************************************/
moveto(65*x,11*y); lineto(81*x,11*y);   /* (F,G) */
moveto(65*x,12*y); lineto(70*x,12*y);   /* (B,D) */
moveto(25*x,11*y);  lineto(55*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***********************************************************************/
moveto(65*x,13*y); lineto(81*x,13*y);   /* (G,H) */
moveto(65*x,14*y); lineto(70*x,14*y);   /* (A,C) */
moveto(25*x,5*y);  lineto(55*x,5*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***********************************************************************/
outtextxy(65*x,18*y,"We are done.");
/***********************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);          /* Clean the game field  again */
bar(2*x,13*y,179*x/2,49*y/2);
bar(59*x,7*y/2,179*x/2,49*y/2);
setcolor(forecolor);
moveto(25*x,5*y); lineto(25*x,11*y);
moveto(45*x,7*y); lineto(35*x,9*y); lineto(45*x,9*y);
moveto(55*x,5*y); lineto(55*x,11*y);
}
```

```
/************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
   switch (ch)      {
    case 'y': in_the_exercise = 0;
          break;
    case 'Y': in_the_exercise = 0;
          break;

    case 'n': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    case 'N': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    default : break;
    }
}
```

```
/* PROGRAM  : binary.c
  AUTHOR    : Atilla BAKAN
  DATE      : Feb. 14, 1990
  REVISED   : Apr. 17, 1990

  DESCRIPTION : This program contains the tutorial for binary trees.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/
/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                   /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)     _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)    _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name        name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name        name
   #define ff_attrib      attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void add_shadow    (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit    (int errnum);
static void initialize    (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void Pageup        (void);
static void Pagedown      (void);
static void press_a_key   (int wrow);
static void pre_help      (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);

/* tutorial functions     */
static void binary_trees  (void);
static void definition_4_5_1 (void);
static void ex_binary_1   (void);
static void ex_binary_2   (void);
static void ex_binary_3   (void);
static void ex_binary_4   (void);
static void ex_binary_5   (void);
static void ex_binary_6   (void);
static void ex_binary_7   (void);
static void ex_binary_8   (void);
static void ex_binary_9   (void);
static void ex_binary_10  (void);
static void expression    (void);
static void preorder      (void);
static void postorder (void);
static void inorder    (void);
static void P1          (void);
static void P2          (void);
static void P3          (void);
static void P4          (void);
static void P5          (void);
static void P6          (void);
static void P7          (void);
static void P8          (void);
static void P9          (void);
static void P10         (void);
static void P11         (void);
static void P12         (void);
static void P13         (void);
static void P14         (void);
static void P15         (void);
static void P16         (void);
static void P17         (void);
static void P18         (void);
static void P19         (void);
static void P20         (void);
static void P21         (void);
static void P22         (void);
static void P23         (void);
static void P24         (void);
static void P25         (void);
static void P26         (void);
static void P27         (void);
static void P28         (void);
static void P29         (void);
static void P30         (void);
static void polish      (void);
static void exercises (void);
static void exer1       (void);
static void exer2       (void);
static void exer3       (void);
static void exer4       (void);
static void exer5       (void);
static void exer6       (void);
static void exer7       (void);
static void exer8       (void);
static void exer9       (void);
static void exer10      (void);
static void exer11      (void);
static void exer12      (void);

/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;

/**********************************************************************/
/* error message table                                                */
/**********************************************************************/
static char *error_text[]= {
   NULL,  /* errnum =  0, no error    */
   NULL,  /* errnum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
```

```
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
      "\t -m = force monochrome text attributes",
    "Memory allocation error"
};

/************************************************************************/
/* miscellaneous defines                                                */
/************************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33

/************************************************************************/
/* this function will add a shadow to the active window                 */
/************************************************************************/
static void add_shadow(void)
{
    wshadow(LGREY|_BLACK);
}

/************************************************************************/
/* this function pops open a window and confirms that the user really   */
/* wants to quit the demo.  If so, it terminates the demo program.      */
/************************************************************************/
static void confirm_quit(void)
{
    struct _onkey_t *kblist;

    kblist=chgonkey(NULL);  /* hide any existing hot keys */
    if(_mouse&MS_CURS) mshidecur();
    if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
    add_shadow();
    wputs("\n Quit demo, are you sure? \033A\156Y\b");
    clearkeys();
    showcur();
    if(wgetchf("YN",'Y')=='Y') normal_exit();
```

```
   wclose();
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}

/**********************************************************************/
/* this function is called by the pull-down demo for a prompt              */
/**********************************************************************/
static void disp_sure_msg(void)
{
   wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
/**********************************************************************/
//* this function handles abnormal termination.  If it is passed an        */
/* error code of 1, then it is a windowing system error.  Otherwise        */
/* the error message is looked up in the error message table.             */
/**********************************************************************/
static void error_exit(int errnum)
{
   if(errnum) {
      printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
               exit(errnum);
   }
}
/**********************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems      */
/**********************************************************************/
static void initialize(void)
{
   /* initialize the CXL video system and save current screen info */
   videoinit();
   readcur(&crow,&ccol);
   if((savescrn=ssave())==NULL) error_exit(3);
   /* if mouse exists, turn on full mouse support */
   if(msinit()) {
        mssupport(MS_FULL);
        msgotoxy(12,49);
    }
    /* attach [Alt-X] to the confirm_quit() function */
    setonkey(0x2d00,confirm_quit,0);

    /* attach [Ctrl Pageup] to the Pageup() function */
    setonkey(0x8400,Pageup,0);

    /* attach [Ctrl Pagedown] to the Pagedown() function */
    setonkey(0x7600,Pagedown,0);

    /* initialize help system, help key = [F1] */
    whelpdef("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
             WHITE|_RED,RED|_LGREY ,pre_help);
}
/*************************************************************************/
/* this function is called anytime to switch back to previous window.   */
/*************************************************************************/
static void Pageup(void)
{
    static WINDOW handle;

    handle = whandle();
    wactiv(handle - 1);
}
/*************************************************************************/
/* this function is called anytime to switch back to next window.       */
/*************************************************************************/
static void Pagedown(void)
{
    static WINDOW handle;

    handle = whandle();
    wactiv(handle + 1);
}
```

```
/*************************************************************************/
static void pre_help(void)
{
   add_shadow();
   setonkey(0x2d00,confirm_quit,0);
}

/*************************************************************************/
/* this function handles normal termination. The original screen and cursor      */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.              */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
/*************************************************************************/
/* this function displays a pause message then pauses for a keypress             */
/*************************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;

   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"           ");
   wprints(wrow,0,attr2,"         ");
}
```

```
/*************************************************************************/
/* This routine causes short dealys during execution                     */
/*************************************************************************/
static void short_delay(void)
{
  delay_(SHORT_DELAY);
}


/*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.             */
/*************************************************************************/
static void quit_window(void)
{
  static WINDOW handle=0;

  if(handle) {
    wactiv(handle);
    wclose();
    handle=0;
  }
  else {
    handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
    wputs(" Quit takes you back to the\n demo program's main menu.");
  }
}

/*************************************************************************/
/* shows the cursor again if it has been hidden                          */
/*************************************************************************/
static void restore_cursor(void)
{
  wtextattr(WHITE|_MAGENTA);
  showcur();
}
```

```
/*********************************************************************/
/* enlarges or shrinks the windows                                   */
/*********************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}

/*********************************************************************/
/* moves the active window to a given screen coordinates             */
/*********************************************************************/
static void move_window(int nsrow,int nscol)
{
   if(wmove(nsrow,nscol)) error_exit(1);
   short_delay();
}
/*********************************************************************/
/* this routine  calls binary_trees() routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                 */
/*********************************************************************/
void P1()
{
   wcloseall();
   binary_trees();
}

/*********************************************************************/
/* this routine  calls definition 4-5-1 routine whenever Pageup or   */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P2()
{
   wcloseall();
   definition_4_5_1();
}
```

```
/**********************************************************************/
/* this routine  calls ex_binary_1 routine whenever Pageup or        */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P3()
{
  wcloseall();
  ex_binary_1();
}
/**********************************************************************/
/* this routine calls ex_binary_2 routine whenever Pageup or         */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P4()
{
  wcloseall();
  ex_binary_2();
}
/**********************************************************************/
/* this routine  calls ex_binary_3 routine whenever Pageup or        */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P5()
{
  wcloseall();
  ex_binary_3();
}
/**********************************************************************/
/* this routine calls preorder routine whenever Pageup or            */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P6()
{
  wcloseall();
  preorder();
}
```

```
/*************************************************************************/
/* this routine calls ex_binary_4 routine whenever Pageup or             */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P7()
{
  wcloseall();
  ex_binary_4();
}
/*************************************************************************/
/* this routine  calls ex_binary_5 routine whenever Pageup or            */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P8()
{
  wcloseall();
  ex_binary_5();
}
/*************************************************************************/
/* this routine  calls polish routine whenever Pageup or                 */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P9()
{
  wcloseall();
  polish();
}
/*************************************************************************/
/* this routine  calls ex_binary_6 routine whenever Pageup or            */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P10()
{
  wcloseall();
  ex_binary_6();
}
```

```
/*************************************************************************/
/* this routine  calls postorder routine whenever Pageup or              */
/* Pagedown keys are pressed.                                            */
/******************************************* ************************/
void P11()
{
  wcloseall();
  postorder();
}
/*************************************************************************/
/* this routine   calls ex_binary_7 routine whenever Pageup or           */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P12()
{
  wcloseall();
  ex_binary_7();
}
/*************************************************************************/
/* this routine  calls ex_binary_8 routine whenever Pageup or            */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P13()
{
  wcloseall();
  ex_binary_8();
}
/*************************************************************************/
/* this routine   calls inorder routine whenever Pageup or               */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P14()
{
  wcloseall();
  inorder();
}
```

```
/**********************************************************************/
/* this routine calls ex_binary_9 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P15()
{
  wcloseall();
  ex_binary_9();
}
/**********************************************************************/
/* this routine  calls ex_binary_10 routine whenever Pageup or        */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P16()
{
  wcloseall();
  ex_binary_10();
}
/**********************************************************************/
/* this routine  calls exercises routine whenever Pageup or           */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P17()
{
  wcloseall();
  exercises();
}
/**********************************************************************/
/* this routine  calls exer1 routine whenever Pageup or               */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P18()
{
  wcloseall();
  exer1();
}
```

```
/***********************************************************************/
/* this routine  calls exer2 routine whenever Pageup or                */
/* Pagedown keys are pressed.                                          */
/***********************************************************************/
void P19()
{
  wcloseall();
  exer2();
}
/***********************************************************************/
/* this routine  calls exer3 routine whenever Pageup or                */
/* Pagedown keys are pressed.                                          */
/***********************************************************************/
void P20()
{
  wcloseall();
  exer3();
}
/***********************************************************************/
/* this routine calls exer4 routine whenever Pageup or                 */
/* Pagedown keys are pressed.                                          */
/***********************************************************************/
void P21()
{
  wcloseall();
  exer4();
}
/***********************************************************************/
/* this routine  calls exer5 routine whenever Pageup or                */
/* Pagedown keys are pressed.                                          */
/***********************************************************************/
void P22()
{
  wcloseall();
  exer5();
}
```

```
/**********************************************************************/
/* this routine  calls exer6 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P23()
{
   wcloseall();
   exer6();
}
/**********************************************************************/
/* this routine  calls exer7 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P24()
{
   wcloseall();
   exer7();
}
/**********************************************************************/
/* this routine calls exer8 routine whenever Pageup or               */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P25()
{
   wcloseall();
   exer8();
}
/**********************************************************************/
/* this routine calls exer9 routine whenever Pageup or               */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P26()
{
   wcloseall();
   exer9();
}
```

```
/*********************************************************************/
/* this routine calls exer10 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P27()
{
   wcloseall();
   exer10();
}
/*********************************************************************/
/* this routine  calls exer11 routine whenever Pageup or             */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P28()
{
   wcloseall();
   exer11();
}
/*********************************************************************/
/* this routine  calls exer12 routine whenever Pageup or             */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P29()
{
   wcloseall();
   exer12();
}
/*********************************************************************/
/* this routine  calls expression routine whenever Pageup or         */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P30()
{
   wcloseall();
   expression();
}
```

```
/***********************************************************************/
/* main routine  calls minimal spanning tree tutorial                  */
/***********************************************************************/
void main()
{
  initialize();
  binary_trees();
}

/***********************************************************************/
/* This routine  calls definition, example and algorithm routines about */
/* binary trees.                                                        */
/***********************************************************************/
static void binary_trees(void)
{
   cclrscrn(LGREY|_BLUE);
   /*******************************************************************/
   /* attach [Pagedown] to the definition_4_5_1() function         */
   setonkey(0x5100,P2,0);
   /*******************************************************************/
   if((w[1]=wopen(5,15,11,65,3,LCYAN|_GREEN,BLACK|_MAGENTA))==0)
            error_exit(1);
   wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputsw("  In previous examples and applications of rooted trees "
            " we need not distinguish between the children of a parent. "
            " However there are many situations where we need this"
           "  distinction.");
   press_a_key(4);
   wslide(1,1);
   /*******************************************************************/
   if((w[2]=wopen(5,15,15,65,3,LCYAN|_GREEN,BLACK|_LGREY))==0)
            error_exit(1);
   wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
   add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  Consider  A/B. In this particular case the order of A and B"
            " is important. Thus if we represent A/B by a rooted tree in"
            " which the root represents the operation (/) and the children"
            " represent the operants (A and B), then the order of the"
           " children is important.");
  wputs("\n");
  wputsw("  This is why we need binary trees to be able to work with"
            " this kind of situations.");
  press_a_key(8);
  wslide(10,1);
  /*********************************************************************/
  short_delay();
  definition_4_5_1();
}

/***********************************************************************/
/* This routine gives the definition of a binary tree and concepts related             */
/* with binary trees.                                                                  */
/***********************************************************************/
static void definition_4_5_1(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the binary_trees() function */
  setonkey(0x4900,P1,0);
  /*********************************************************************/
  /* attach [Pagedown] to the ex_binary_1() function             */
  setonkey(0x5100,P3,0);
  /*********************************************************************/
  if((w[3]=wopen(8,20,12,60,3,LCYAN|_GREEN,BLACK|_RED))==0)
            error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n          What is a binary tree ?");
  press_a_key(2);
```

```
    wcloseall();
    short_delay();
    /**************************************************************************/
    if((w[4]=wopen(5,15,17,65,3,LCYAN|_GREEN,BLACK|_LGREY))==0)
            error_exit(1);
    wtitle("[Definition - Binary Trees]",TCENTER,_LGREY|BROWN);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputsw("   A binary tree is a rooted tree in which each vertex has at "
           "  most two children and each child is designated as being"
           " a left child or a right child.");
    wputs("\n");
    wputsw("   The left subtree of a vertex V in a binary tree is the "
           " graph formed by the left child of V, the descendants of L,"
           "  and the edges connecting these vertices. ");
    wputs("\n");
    wputsw("   The right subtree of a vertex V in a binary tree is  "
           " defined in the same manner. ");
    press_a_key(10);
    short_delay();
    wclose();
    ex_binary_1();
}
```

```
/***************************************************************************/
/* This routine gives an example for a typical binary tree                */
/***************************************************************************/
static void ex_binary_1 (void)
{
  /*************************************************************************/
  /* attach [Pageup] to the definition_4_5_1() function */
  setonkey(0x4900,P2,0);
  /*************************************************************************/
  /* attach [Pagedown] to the expression() function        */
  setonkey(0x5100,P30,0);
  /*************************************************************************/
  if((w[5]=wopen(8,20,13,60,3,LCYAN|_GREEN,RED|_BLACK))==0)
          error_exit(1);
  wtitle("[Binary Trees -Example 1]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n          To see an example ");
  press_a_key(3);
  short_delay();
  wcloseall();
  /*************************************************************************/
  spawnl(P_WAIT,"exb1.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  expression();
}
```

```
/*************************************************************************/
/* This routine tells about expression trees                            */
/*************************************************************************/
static void expression(void)
{
  /**********************************************************************/
  /* attach [Pageup] to the ex_binary_1() function */
  setonkey(0x4900,P3,0);
  /**********************************************************************/
  /* attach [Pagedown] to the ex_binary_2() function           */
  setonkey(0x5100,P4,0);
  /**********************************************************************/
  if((w[6]=wopen(5,15,13,54,3,WHITE|_RED,BLACK|_LGREY))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" There are many applications of  binary trees in computer"
         " science. Such as repesenting ways to organize data and"
         " describe data. One peculiar application is known as"
         " 'expression tree'. ");
  press_a_key(6);
  wslide(0,0);
  if((w[7]=wopen(12,1,20,39,3,RED|_BLACK,WHITE|_BLUE))==0) error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  An expression tree is a binary tree which is used for "
         "  representing arithmetic expressions. In this binary tree "
         "  operations are represented as internal vertices and the "
         "  operands as terminal vertices.");
  press_a_key(6);
  wslide(0,40);
  short_delay();
  ex_binary_2();
 }
```

```
/*********************************************************************/
/* This routine gives an example for an expression tree              */
/*********************************************************************/
static void ex_binary_2 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the expression() function */
  setonkey(0x4900,P30,0);
  /*******************************************************************/
  /* attach [Pagedown] to the ex_binary_3() function           */
  setonkey(0x5100,P5,0);
  /*******************************************************************/
  if((w[8]=wopen(12,20,17,60,3,LCYAN|_GREEN,RED|_BLACK))==0)
          error_exit(1);
  wtitle("[Binary Trees - Example 2]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n        How about an example ? ");
  press_a_key(3);
  short_delay();
  wcloseall();
  /*******************************************************************/
  spawnl(P_WAIT,"exb2.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  ex_binary_3();
}
```

```
/**********************************************************************/
/* This routine gives a somewhat complicated example for an expression tree    */
/**********************************************************************/
static void ex_binary_3 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the ex_binary_2() function */
  setonkey(0x4900,P4,0);
  /* attach [Pagedown] to the preorder() function          */
  setonkey(0x5100,P6,0);
  /*******************************************************************/
  if((w[9]=wopen(12,20,17,60,3,LCYAN|_GREEN,RED|_BLACK))==0)
            error_exit(1);
  wtitle("[Binary Trees - Example 3]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n How about a more complicated example ?");
  press_a_key(3);
  short_delay();
  wslide(5,20);
  if((w[9]=wopen(12,20,18,60,3,LCYAN|_BLACK,BLACK|_RED))==0)
            error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" How would you represent the following expression as a"
     " binary tree ?");
  wputs("\n(((6 - 3) * 2) + 7) / ((5 - 1) * 4 + 8)");
  press_a_key(4);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"exb3.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  preorder();
}
```

```
/***********************************************************************/
/* This routine teaches the preorder traversal in binary trees          */
/***********************************************************************/
static void preorder(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the ex_binary_3() function        */
  setonkey(0x4900,P5,0);
  /* attach [Pagedown] to the ex_binary_4() function */
  setonkey(0x5100,P7,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,15,54,3,LCYAN|_GREEN,BLACK|_LGREY))==0)
           error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  So far, we have shown you how an arithmetic expression"
          " can be represented by an expression tree. It's okay, but"
          " we somehow need to process the expression tree to obtain"
          " the original expression.");
  press_a_key(8);
  wslide(0,0);
  short_delay();
  if((w[2]=wopen(5,15,15,54,3,LCYAN|_GREEN,WHITE|_BLACK))==0)
           error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  We need a systematic way to look at each vertex in the "
          " expression  tree exactly once. Processing the data at a "
          " vertex is usually called 'visiting a vertex', and a search"
          " procedure that visits each vertex of a graph exactly once "
          " is called a 'traversal' of a graph.");
  press_a_key(8);
  wslide(0,39);
  short_delay();
```

```
    if((w[3]=wopen(5,15,19,54,3,LCYAN|_GREEN,WHITE|_RED))==0)
             error_exit(1);
    wtitle("[Preorder Traversal]",TCENTER,_LGREY|BROWN);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputsw("  One traversal method is known as Preorder Traversal."
            " This method is simply a depth-first search to a binary"
            " tree starting at the root and always choosing a left "
           " child of a vertex when there is a choise. We consider"
           " a vertex visited when it is labeled, and keep a list"
           " of the vertices in the order visited. This list is"
          " called a preorder listing. Actual algorithm is as"
         " follows.");
    press_a_key(12);
    wslide(11,0);
    short_delay();
    if((w[4]=wopen(5,15,19,54,3,LCYAN|_GREEN,BLACK|_CYAN))==0)
             error_exit(1);
    wtitle("[Preorder Traversal]",TCENTER,_LGREY|BROWN);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputsw("  The following algorithm is a recursive formulation of"
            " the preorder traversal, which means that in this description"
            " the algorithm refers itself.");
    wputs("\n");
    wputs("    Step 1 (visit)    Visit the root.\n");
    wputsw("    Step 2 (go left)  Go to the left subtree, if one exists,"
            " do a preorder traversal.");
    wputs("\n");
    wputsw("    Step 3 (go right) Go to the right subtree, if one exists,"
            " and do a preorder traversal.");
    press_a_key(12);
    wslide(11,39);
    short_delay();
    ex_binary_4();
}
```

```
/**********************************************************************/
/* This routine gives a preorder traversal of a binary tree           */
/**********************************************************************/
static void ex_binary_4 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the preorder() function          */
  setonkey(0x4900,P6,0);
  /*******************************************************************/
  /* attach [Pagedown] to the ex_binary_5() function */
  setonkey(0x5100,P8,0);
  /*******************************************************************/
  if((w[5]=wopen(8,20,13,60,3,LCYAN|_GREEN,RED|_BLACK))==0)
          error_exit(1);
  wtitle("[Preorder Traversal - Example 4]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n        How about an example ? ");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"exb4.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  ex_binary_5();
}
```

```
/**********************************************************************/
/* This routine gives a preorder traversal of a binary tree          */
/**********************************************************************/
static void ex_binary_5 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the ex_binary_4() function        */
  setonkey(0x4900,P7,0);
  /*******************************************************************/
  /* attach [Pagedown] to the polish() function */
  setonkey(0x5100,P9,0);
  /*******************************************************************/
  if((w[6]=wopen(8,20,13,60,3,LCYAN|_GREEN,RED|_BLACK))==0)
          error_exit(1);
  wtitle("[Preorder Traversal - Example 5]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n      One more example ? ");
  press_a_key(3);
  short_delay();
  wclose();
  spawnl(P_WAIT,"exb5.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  polish();
}
```

```
/**********************************************************************/
/* This routine introduces prefix form ( Polish notation )            */
/**********************************************************************/
static void polish(void)
{
  /********************************************************************/
  /* attach [Pageup] to the ex_binary_5() function           */
  setonkey(0x4900,P8,0);
  /********************************************************************/
  /* attach [Pagedown] to the ex_binary_6() function */
  setonkey(0x5100,P10,0);
  /********************************************************************/
  if((w[1]=wopen(5,15,14,56,3,LCYAN|_GREEN,BLACK|_LGREY))==0)
          error_exit(1);
  wtitle("[Polish Notation]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  When a preorder traversal is applied on an expression"
         " tree, the resulting listing of operations and operands is"
        " called prefix form or Polish notation for the expression."
       " (the later name is used in honor of the famous Polish"
       " logician Lukasiewicz.)");
  press_a_key(7);
  wslide(0,0);
  short_delay();
  /********************************************************************/
  if((w[2]=wopen(5,15,17,53,3,LCYAN|_GREEN,WHITE|_BLACK))==0)
          error_exit(1);
  wtitle("[Polish Notation]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" An expression in Polish notation is evaluated according"
         " to the following rule :  Scan from left to right until coming"
         " to an operation sign, say T, that is followed by by two "
         " successive numbers, say a and b. Evaluate Tab as aTb, and"
        " replace Tab by this value in the expression. Repeat this"
```

```
            " process until the entire expression is evaluated.");
  press_a_key(10);
  wslide(0,39);
  short_delay();
  ex_binary_6();
}
/**************************************************************************/
/* This routine gives an example on Polish  notation of an expression          */
/**************************************************************************/
static void ex_binary_6 (void)
{
  /***********************************************************************/
  /* attach [Pageup] to the polish() function              */
  setonkey(0x4900,P9,0);
  /***********************************************************************/
  /* attach [Pagedown] to the postorder() function */
  setonkey(0x5100,P11,0);
  /***********************************************************************/
  if((w[3]=wopen(12,12,22,63,3,LCYAN|_BLACK,BLACK|_RED))==0)
            error_exit(1);
  wtitle("[Polish Notation - Example 6]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n   Do you remember the following expression ?");
  wputs("\n     (((6 - 3) * 2) + 7) / ((5 - 1) * 4 + 8)\n\n");
  wputsw("   Now let's see how we represent this expression in Polish"
            " notation and then we will show you the application of the"
            " rule for evaluating this the expression written in this"
            " notation");
  press_a_key(8);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"exb6.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  postorder();
}
```

```
/**********************************************************************/
/* This routine teaches the postorder traversal in binary trees       */
/**********************************************************************/
static void postorder(void)
{
  /********************************************************************/
  /* attach [Pageup] to the ex_binary_6() function          */
  setonkey(0x4900,P10,0);
  /********************************************************************/
  /* attach [Pagedown] to the ex_binary_7() function */
  setonkey(0x5100,P12,0);
  /********************************************************************/
  if((w[1]=wopen(5,13,13,56,3,LCYAN|_GREEN,BLACK|_LGREY))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" The Polish notation for an expression provides an unambiguous"
         " way to write it without the use of paranthesis or conventions"
        " about the order of operations. Many computers are designed to"
       " rewrite expression in this form.");
  press_a_key(6);
  wclose();
  short_delay();
  /********************************************************************/
  if((w[2]=wopen(5,15,15,54,3,LCYAN|_GREEN,WHITE|_BLACK))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" Second traversal method that we are about to examine"
         " is known as reverse Polish notation or postfix form."
        " This method is also introduced by Lukasiewicz. Unlike"
        " Polish notation, in this method the operation sign"
       " follows the operands.");
  press_a_key(8);
```

```
wslide(0,0);
short_delay();
/*******************************************************************/
if((w[3]=wopen(5,15,19,54,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
wtitle("[Postorder Traversal]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw(" Reverse Polish notation for the last expression that we"
         " examined");
wputs("\n(((6 - 3) * 2) + 7) / (((5 - 1) * 4) + 8)\n");
wputs("\n is '6 3 - 2 * 7 + 5 1 - 4 * 8 + /'\n\n");
wputsw(" Again as you see we did not use paranthesis that is we"
         " don't worry about the order of the expressions. Thus reverse"
         " Polish is an efficient method for use in computers. ");
press_a_key(12);
wslide(11,0);
short_delay();
/*******************************************************************/
if((w[4]=wopen(5,15,13,54,3,LCYAN|_GREEN,WHITE|_BLACK))==0)
          error_exit(1);
wtitle("[Postorder Traversal]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw(" How can the reverse Polish notation for an expression"
         " be obtained from an expression tree ?");
wputs("\n");
wputsw(" We bet you are asking this question now. Here how"
         " it is ...");
press_a_key(6);
wslide(0,39);
short_delay();
/*******************************************************************/
if((w[5]=wopen(5,15,18,54,3,LCYAN|_GREEN,BLACK|_CYAN))==0)
          error_exit(1);
wtitle("[Postorder Traversal]",TCENTER,_LGREY|BROWN);
```

```
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" The following algorithm is a recursive formulation of"
         " the postorder traversal");
  wputs("\n");
  wputsw("   Step 1 (go left)  Go to the left subtree, if one exists,"
         " do a postorder traversal.");
  wputs("\n");
  wputsw("   Step 2 (go right) Go to the right subtree, if one exists,"
         " and do a postorder traversal.");
  wputs("\n   Step 3 (visit)   Visit the root.\n");
  press_a_key(12);
  wslide(11,39);
  short_delay();
  ex_binary_7();
}

/*********************************************************************/
/* This routine gives a postorder traversal of a binary tree         */
/*********************************************************************/
static void ex_binary_7 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the postorder() function           */
  setonkey(0x4900,P11,0);
  /*******************************************************************/
  /* attach [Pagedown] to the ex_binary_8() function */
  setonkey(0x5100,P13,0);
  /*******************************************************************/
           if((w[6]=wopen(8,18,13,62,3,LCYAN|_GREEN,RED|_BLACK))==0)    er-
ror_exit(1);
  wtitle("[Postorder Traversal- Example 7]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n We need an example, don't you think so ? ");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"exb7.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  ex_binary_8();
}




/**********************************************************************/
/* This routine gives a postorder traversal of a binary tree          */
/**********************************************************************/
static void ex_binary_8 (void)
{
  /********************************************************************/
  /* attach [Pageup] to the ex_binary_7() function          */
  setonkey(0x4900,P12,0);
  /********************************************************************/
  /* attach [Pagedown] to the inorder() function */
  setonkey(0x5100,P14,0);
  /********************************************************************/
        if((w[7]=wopen(8,20,13,60,3,LCYAN|_GREEN,RED|_BLACK))==0)   er-
ror_exit(1);
  wtitle("[Postorder traversal - Example 8]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n      One more example ? ");
  press_a_key(3);
  short_delay();
  wclose();
  spawnl(P_WAIT,"exb8.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  inorder();
}
```

```
/***********************************************************************/
/* This routine teaches the inorder traversal in binary trees          */
/***********************************************************************/
static void inorder(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the ex_binary_8() function              */
  setonkey(0x4900,P13,0);
  /* attach [Pagedown] to the ex_binary_9() function */
  setonkey(0x5100,P15,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,14,54,3,LCYAN|_GREEN,BLACK|_RED))==0)
        error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" We have seen how expression trees are converted"
        " into Polish and reverse Polish notations for an"
        " expression. In these notations the operation sign"
        " precedes or follows the operands, respectively.");
  press_a_key(7);
  wslide(0,0);
  short_delay();
  /*********************************************************************/
  if((w[2]=wopen(5,15,14,54,3,LCYAN|_GREEN,WHITE|_BLACK))==0)
        error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" Make a guess, where else can we put operation"
        " sign. The answer should be so difficult, because"
        " there is only one place left, that is , the operation"
        " sign can be in between two operants. This type is the"
        " one we accustomed to see. ");
  press_a_key(7);
  wslide(0,39);
```

```
    short_delay();
    /**********************************************************************/
    if((w[3]=wopen(5,15,14,54,3,LCYAN|_GREEN,RED|_BLACK))==0)
              error_exit(1);
    wtitle("[Inorder Traversal]",TCENTER,_LGREY|BROWN);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputsw(" With the use of the inorder traversal it is possible"
              " to obtain an expression with the operation sign between"
             " the operands. However, this traversal requires the"
             " careful insertion of paranthesis in order to evaluate"
             " the expression properly.");
    press_a_key(7);
    wslide(10,0);
    short_delay();
    /**********************************************************************/
    if((w[4]=wopen(5,15,19,54,3,LCYAN|_GREEN,BLACK|_CYAN))==0)
              error_exit(1);
    wtitle("[Inorder Traversal]",TCENTER,_LGREY|BROWN);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputsw(" The Inorder Traversal is characterized by visiting a"
              " left child before the parent and a right child after"
              " the parent. The following algorithm shows the systematic"
              " way  to do this.");
    wputs("\n");
    wputsw("    Step 1 (go left)  Go to the left subtree, if one exists,"
              " do an inorder traversal.");
    wputs("\n    Step 2 (visit)    Visit the root.\n");
    wputsw("    Step 3 (go right) Go to the right subtree, if one exists,"
              " and do an inorder traversal.");
    press_a_key(13);
    wslide(10,39);
    short_delay();
    ex_binary_9();
}
```

```
/************************************************************************/
/* This routine gives an inorder traversal of a binary tree                */
/************************************************************************/
static void ex_binary_9 (void)
{
  /**********************************************************************/
  /* attach [Pageup] to the inorder() function          */
  setonkey(0x4900,P14,0);
  /**********************************************************************/
  /* attach [Pagedown] to the ex_binary_10() function */
  setonkey(0x5100,P16,0);
  /**********************************************************************/
  if((w[5]=wopen(8,18,13,62,3,LCYAN|_GREEN,RED|_BLACK))==0)
          error_exit(1);
  wtitle("[Inorder Traversal - Example 9]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n       Let' see  an example... ");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"exb9.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  ex_binary_10();
}
```

```
/*********************************************************************/
/* This routine gives an inorder traversal of a binary tree         */
/*********************************************************************/
static void ex_binary_10 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the ex_binary_9() function          */
  setonkey(0x4900,P15,0);
  /*******************************************************************/
  /* attach [Pagedown] to the exercises() function */
  setonkey(0x5100,P17,0);
  /*******************************************************************/
  if((w[6]=wopen(8,20,13,60,3,LCYAN|_GREEN,RED|_BLACK))==0)
            error_exit(1);
  wtitle("[Inorder Traversal - Example 10]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n          One more example ? ");
  press_a_key(3);
  short_delay();
  wclose();
  spawnl(P_WAIT,"exb10.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exercises();
}
```

```
/***************************************************************************/
/* This routine makes a small quiz about the binary trees and travesals.        */
/***************************************************************************/
void exercises(void)
 {
   register int *screen;

   /***********************************************************************/
   /* attach [Pageup] to the ex_binary_10() function              */
   setonkey(0x4900,P16,0);
   /***********************************************************************/
   /* attach [Pagedown] to the exer1() function */
   setonkey(0x5100,P18,0);
   /***********************************************************************/
   if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
             error_exit(1);
   wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   add_shadow();
   wputs("\n");
   wputsw(" We have completed our presentation of this section. Are"
            " you ready for a pop quiz ? ");
   press_a_key(3);
   short_delay();
   wclose();
   if((screen=ssave())==NULL) error_exit(3);
   exer1();
    /* if mouse exists, turn on full mouse support */
   if(msinit()) {
     mssupport(MS_FULL);
     msgotoxy(12,49);
    }
   /* attach [Alt-X] to the confirm_quit() function */
   setonkey(0x2d00,confirm_quit,0);
   srestore(screen);
 }
```

```
/*********************************************************************/
/* Dummy function to call the actual exercise 4.5.1                  */
/*********************************************************************/
static void exer1(void)
{
   /*****************************************************************/
   /* attach [Pageup] to the ex_binary_10() function          */
   setonkey(0x4900,P16,0);
   /*****************************************************************/
   /* attach [Pagedown] to the exer2() function */
   setonkey(0x5100,P19,0);
   /*****************************************************************/
   if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
             error_exit(1);
   wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   add_shadow();
   wputs("\n");
   wputsw("         Here is the first question. ");
   press_a_key(3);
   wclose();
   spawnl(P_WAIT,"q451.exe",NULL);
   cclrscrn(LGREY|_BLUE);
   exer2();
}
```

```
/****************************************************************************/
/* Dummy function to call the actual exercise 4.5.2                         */
/****************************************************************************/
static void exer2(void)
{
  /**************************************************************************/
  /* attach [Pageup] to the exer1() function             */
  setonkey(0x4900,P18,0);
  /**************************************************************************/
  /* attach [Pagedown] to the exer3() function */
  setonkey(0x5100,P20,0);
  /**************************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the second question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q452.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer3();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.5.3                     */
/*************************************************************************/
static void exer3(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the exer2() function              */
  setonkey(0x4900,P19,0);
  /***********************************************************************/
  /* attach [Pagedown] to the exer4() function */
  setonkey(0x5100,P21,0);
  /***********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
            error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the third question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q453.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer4();
}
```

```
/**********************************************************************/
/* Dummy function to call the actual exercise 4.5.4                  */
/**********************************************************************/
static void exer4(void)
{
  /********************************************************************/
  /* attach [Pageup] to the exer3() function         */
  setonkey(0x4900,P20,0);
  /********************************************************************/
  /* attach [Pagedown] to the exer5() function */
  setonkey(0x5100,P22,0);
  /********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the forth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q454.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer5();
}
```

```
/**********************************************************************/
/* Dummy function to call the actual exercise 4.5.5                   */
/**********************************************************************/
static void exer5(void)
{
  /********************************************************************/
  /* attach [Pageup] to the exer4() function         */
  setonkey(0x4900,P21,0);
  /********************************************************************/
  /* attach [Pagedown] to the exer6() function */
  setonkey(0x5100,P23,0);
  /********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the fifth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q455.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer6();
}
```

```
/**************************************************************************/
/* Dummy function to call the actual exercise 4.5.6                        */
/**************************************************************************/
static void exer6(void)
{
  /************************************************************************/
  /* attach [Pageup] to the exer5() function           */
  setonkey(0x4900,P22,0);
  /************************************************************************/
  /* attach [Pagedown] to the exer7() function */
  setonkey(0x5100,P24,0);
  /************************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
            error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the sixth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q456.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer7();
}
```

```
/*********************************************************************/
/* Dummy function to call the actual exercise 4.5.7                   */
/*********************************************************************/
static void exer7(void)
{
  /*******************************************************************/
  /* attach [Pageup] to the exer6() function           */
  setonkey(0x4900,P23,0);
  /*******************************************************************/
  /* attach [Pagedown] to the exer8() function */
  setonkey(0x5100,P25,0);
  /*******************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the seventh question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q457.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer8();
}
```

```
/**********************************************************************/
/* Dummy function to call the actual exercise 4.5.8                  */
/**********************************************************************/
static void exer8(void)
{
  /********************************************************************/
  /* attach [Pageup] to the exer7() function                         */
  setonkey(0x4900,P24,0);
  /********************************************************************/
  /* attach [Pagedown] to the exer9() function */
  setonkey(0x5100,P26,0);
  /********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the eighth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q458.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer9();
}
```

```
/************************************************************************/
/* Dummy function to call the actual exercise 4.5.9                    */
/************************************************************************/
static void exer9(void)
{
  /**********************************************************************/
  /* attach [Pageup] to the exer8() function           */
  setonkey(0x4900,P25,0);
  /**********************************************************************/
  /* attach [Pagedown] to the exer10() function */
  setonkey(0x5100,P27,0);
  /**********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the nineth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q459.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer10();
}
```

```
/*********************************************************************/
/* Dummy function to call the actual exercise 4.5.10                */
/*********************************************************************/
static void exer10(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the exer9() function          */
  setonkey(0x4900,P26,0);
  /*********************************************************************/
  /* attach [Pagedown] to the exer11() function */
  setonkey(0x5100,P28,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the tenth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q4510.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer11();
}
```

```
/*********************************************************************/
/* Dummy function to call the actual exercise 4.5.11                 */
/*********************************************************************/
static void exer11(void)
{
  /*******************************************************************/
  /* attach [Pageup] to the exer10() function         */
  setonkey(0x4900,P27,0);
  /*******************************************************************/
  /* attach [Pagedown] to the exer12() function */
  setonkey(0x5100,P29,0);
  /*******************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the eleventh question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q4511.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer12();
}
```

```
/*********************************************************************/
/* Dummy function to call the actual exercise 4.5.12                 */
/*********************************************************************/
static void exer12(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the exer11() function          */
  setonkey(0x4900,P28,0);
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the twelfth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q4512.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  wcloseall();
  normal_exit();
}
```

```
/* PROGRAM   : exb1.c
  AUTHOR     : Atilla BAKAN
  DATE       : Apr. 16, 1990
  REVISED    : Apr. 16, 1990

  DESCRIPTION : First example about binary trees.



  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/



/* header files */
#include <graphics.h>
#include "cxldef.h"



#if defined(__TURBOC__)                     /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)           /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph         (void);



/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used i   including drivers to the executable code  */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/************************************************************************/
/* This fuction initializes the necessary graphical routines           */
/************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**********************************************************************/
  settext();
  /**********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      }
   forecolor = WHITE;
  }
```

```
/***********************************************************************/
/* This function sets the text default values                          */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
 }



/***********************************************************************/
/* main routine  calls graphs routine                                  */
/***********************************************************************/
void main()
{
  graph();
}
```

```
/*********************************************************************/
/* This routine gives examples of trees and some graphs that are not trees.          */
/*********************************************************************/
 void graph(void)
 {
  /*******************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-1");
  /*******************************************************************/
  pieslice(45*x,4*y,0,359,2);   /* Parent      */
  pieslice(35*x,7*y,0,359,2);   /* Left child  */
  pieslice(55*x,7*y,0,359,2);   /* Right child */
  moveto(35*x,7*y);  lineto(45*x,4*y);  lineto(55*x,7*y);
  outtextxy(45*x,7*y/2,"Parent");
  outtextxy(25*x,7*y,"Left");
  outtextxy(25*x,15*y/2,"child");
  outtextxy(58*x,7*y,"Right");
  outtextxy(58*x,15*y/2,"child");
  pieslice(23*x,10*y,0,359,2);
  pieslice(43*x,10*y,0,359,2);
  pieslice(47*x,10*y,0,359,2);
  pieslice(67*x,10*y,0,359,2);
  moveto(23*x,10*y);  lineto(35*x,7*y); lineto(43*x,10*y);
  moveto(47*x,10*y);  lineto(55*x,7*y); lineto(67*x,10*y);
  outtextxy(2*x,15*y,"Now, all the node you see under the left child (including him-
                              self)");
  outtextxy(2*x,16*y,"is the left subtree.");
  outtextxy(2*x,17*y,"Similarly, all the nodes under right child is called th right sub-
                              tree.");
  delay_(36);
  outtextxy(25*x,11*y,"Left subtree");
  outtextxy(50*x,11*y,"Right subtree");
  Pause(30*x,24*y);
 closegraph();
 videoinit();
}
```

```
/* PROGRAM   : exb2.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 16, 1990
   REVISED    : Apr. 17, 1990

   DESCRIPTION : Second example about binary trees.



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions     */
static void graph        (void);


/*/************************************************************************/
/* graphic initialization variables                                      */
/************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;



/************************************************************************/
/* This function is used for including drivers to the executable code   */
/************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**************************************************************************/
/* This fuction initializes the necessary graphical routines              */
/**************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      }
  forecolor = WHITE;
  }
```

```
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
 }



/*************************************************************************/
/* main routine   calls  graphs routine                                  */
/*************************************************************************/
void main()
{
  graph();
 }
```

```
/***********************************************************************/
/* This routine gives examples of trees and some graphs that are not trees.           */
/***********************************************************************/
 void graph(void)
 {
  /*********************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-2");
  /*********************************************************************/
  outtextxy(2*x,3*y,"The expression a - b (where '-' denotes  substraction) is
                       represented below.");
  outtextxy(2*x,4*y,"Note that the operation - is represented by an represented by
                       an internal ");
  outtextxy(2*x,5*y,"vertex and the operands a and b are represented by terminal
                       vertices.");
  pieslice(45*x,10*y,0,359,2);   /* - */
  pieslice(35*x,13*y,0,359,2);   /* a */
  pieslice(55*x,13*y,0,359,2);   /* b */
  moveto(35*x,13*y); lineto(45*x,10*y); lineto(55*x,13*y);
  outtextxy(45*x,18*y/2,"-");
  outtextxy(35*x,27*y/2,"a");
  outtextxy(55*x,27*y/2,"b");
  /*********************************************************************/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
 }
```

```
/* PROGRAM  : exb3.c
  AUTHOR    : Atilla BAKAN
  DATE      : Apr. 16, 1990
  REVISED   : Apr. 17, 1990

  DESCRIPTION : Third example about binary trees.


  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                  /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                    /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
  #define bioskey(a)       _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)      _dos_findnext(a)
  #define ffblk            find_t
  #define ff_name          name
#elif defined(__ZTC__)                   /* Zortech C/C++ */
  #define ffblk            FIND
  #define ff_name          name
  #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions     */
static void graph        (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)          {
    case 'y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
       default : break;
       }
     hidecur();
     if(_mouse&MS_CURS) msshowcur();
     chgonkey(kblist);     /* restore any hidden hot keys */
}
/**********************************************************************/
/* This function sets the text default values                          */
/**********************************************************************/
static void settext(void)
{
    settextstyle(0,0,0);
    setlinestyle(0,4,3);
    settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
    settext();
    outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
    if(waitkey()==ESC) confirm_graph_exit();
  }
/**********************************************************************/
/* main routine  calls graph  routine                                  */
/**********************************************************************/
void main()
{
    graph();
}
```

```
/***********************************************************************/
/* This routine gives  an example of expression trees.          .          */
/***********************************************************************/
 void graph(void)
 {
  /**********************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-3");
  /**********************************************************************/
  outtextxy(16*x,2*y,"(((6 - 3) * 2) + 7) / ((5 - 1) * 4 + 8)");
  outtextxy(2*x,3*y,"Actully we made things a little bit easier by using paranthesis
                    when we were");
  outtextxy(2*x,4*y,"writing the expression. Now foliow our steps to build the ex-
                    pression tree.");
  pieslice(20*x,7*y,0,359,2);   /* / */
  pieslice(10*x,9*y,0,359,2);   /* ((6 - 3) * 2) + 7) */
  pieslice(30*x,9*y,0,359,2);   /* ((5 - 1) * 4 + 8)  */
  moveto(10*x,9*y); lineto(20*x,7*y); lineto(30*x,9*y);
  outtextxy(20*x,13*y/2,"/");
  outtextxy(3*x,19*y/2,"((6-3)*2)+7)");
  outtextxy(23*x,19*y/2,"((5-1)*4+8)");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
  setcolor(forecolor);
  /**********************************************************************/
  pieslice(60*x,7*y,0,359,2);   /* / */
  pieslice(50*x,9*y,0,359,2);   /* + */
  pieslice(70*x,9*y,0,359,2);   /* + */
  moveto(50*x,9*y); lineto(60*x,7*y); lineto(70*x,9*y);
  outtextxy(60*x,13*y/2,"/");
  outtextxy(48*x,9*y,"+");
  outtextxy(72*x,9*y,"+");
```

```
pieslice(45*x,11*y,0,359,2);   /* (6 - 3) * 2 */
pieslice(55*x,11*y,0,359,2);   /* 7 */
pieslice(65*x,11*y,0,359,2);   /* (5 - 1) * 4 */
pieslice(75*x,11*y,0,359,2);   /* 8 */
moveto(45*x,11*y);  lineto(50*x,9*y);  lineto(55*x,11*y);
moveto(65*x,11*y);  lineto(70*x,9*y);  lineto(75*x,11*y);
outtextxy(40*x,23*y/2,"(6-3)*2");
outtextxy(55*x,23*y/2,"7");
outtextxy(60*x,23*y/2,"(5-1)*4");
outtextxy(75*x,23*y/2,"8");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(25*x,13*y,0,359,2);   /* / */
pieslice(15*x,15*y,0,359,2);   /* + */
pieslice(35*x,15*y,0,359,2);   /* + */
moveto(15*x,15*y);  lineto(25*x,13*y);  lineto(35*x,15*y);
outtextxy(25*x,25*y/2,"/");
outtextxy(13*x,15*y,"+");
outtextxy(37*x,15*y,"+");
pieslice(10*x,17*y,0,359,2);   /* * */
pieslice(20*x,17*y,0,359,2);   /* 7 */
pieslice(30*x,17*y,0,359,2);   /* * */
pieslice(40*x,17*y,0,359,2);   /* 8 */
moveto(10*x,17*y);  lineto(15*x,15*y);  lineto(20*x,17*y);
moveto(30*x,17*y);  lineto(35*x,15*y);  lineto(40*x,17*y);
outtextxy(8*x,17*y,"*");
outtextxy(20*x,35*y/2,"7");
outtextxy(28*x,17*y,"*");
outtextxy(40*x,35*y/2,"8");
pieslice(5*x,19*y,0,359,2);   /* (6 - 3) */
pieslice(15*x,19*y,0,359,2);  /*  2      */
pieslice(25*x,19*y,0,359,2);  /* (5 - 1) */
pieslice(35*x,19*y,0,359,2);  /*  4      */
```

```
moveto(5*x,19*y); lineto(10*x,17*y); lineto(15*x,19*y);
moveto(25*x,19*y); lineto(30*x,17*y); lineto(35*x,19*y);
outtextxy(2*x,39*y/2,"(6 - 3)");
outtextxy(15*x,39*y/2,"2");
outtextxy(22*x,39*y/2,"(5 - 1)");
outtextxy(35*x,39*y/2,"4");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************************/
pieslice(65*x,13*y,0,359,2);   /* / */
pieslice(55*x,15*y,0,359,2);   /* + */
pieslice(75*x,15*y,0,359,2);   /* + */
moveto(55*x,15*y); lineto(65*x,13*y); lineto(75*x,15*y);
outtextxy(65*x,25*y/2,"/");
outtextxy(53*x,15*y,"+");
outtextxy(77*x,15*y,"+");
pieslice(50*x,17*y,0,359,2);   /* * */
pieslice(60*x,17*y,0,359,2);   /* 7 */
pieslice(70*x,17*y,0,359,2);   /* * */
pieslice(80*x,17*y,0,359,2);   /* 8 */
moveto(50*x,17*y); lineto(55*x,15*y); lineto(60*x,17*y);
moveto(70*x,17*y); lineto(75*x,15*y); lineto(80*x,17*y);
outtextxy(48*x,17*y,"*");
outtextxy(60*x,35*y/2,"7");
outtextxy(68*x,17*y,"*");
outtextxy(80*x,35*y/2,"8");
pieslice(45*x,19*y,0,359,2);   /* - */
pieslice(55*x,19*y,0,359,2);   /* 2 */
pieslice(65*x,19*y,0,359,2);   /* - */
pieslice(75*x,19*y,0,359,2);   /* 4 */
moveto(45*x,19*y); lineto(50*x,17*y); lineto(55*x,19*y);
moveto(65*x,19*y); lineto(70*x,17*y); lineto(75*x,19*y);
outtextxy(43*x,19*y,"-");
outtextxy(55*x,39*y/2,"2");
```

```
 outtextxy(62*x,19*y,"-");
 outtextxy(75*x,39*y/2,"4");
 pieslice(40*x,21*y,0,359,2);   /* 6 */
 pieslice(50*x,21*y,0,359,2);   /* 3 */
 pieslice(60*x,21*y,0,359,2);   /* 5 */
 pieslice(70*x,21*y,0,359,2);   /* 1 */
 moveto(40*x,21*y); lineto(45*x,19*y); lineto(50*x,21*y);
 moveto(60*x,21*y); lineto(65*x,19*y); lineto(70*x,21*y);
 outtextxy(40*x,43*y/2,"6");
 outtextxy(50*x,43*y/2,"3");
 outtextxy(60*x,43*y/2,"5");
 outtextxy(70*x,43*y/2,"1");
 Pause(30*x,24*y);
/*************************************************************************/
 closegraph();
 videoinit();
}
```

```
/* PROGRAM   : exb4.c
  AUTHOR     : Atilla BAKAN
  DATE       : Apr. 16, 1990
  REVISED    : Apr. 17, 1990

  DESCRIPTION : Third example about binary trees. It gives an example of
                preorder traversal.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph         (void);

/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*********************************************************************/
/* This function is used for including drivers to the executable code */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*********************************************************************/
/* This fuction initializes the necessary graphical routines        */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)          {
     case 'y': closegraph();
           videoinit();
           exit(0);
           break;
     case 'Y': closegraph();
           videoinit();
           exit(0);
           break;
     case 'n': setcolor(backcolor);
           bar(4*x/3,23*y,30*x,97*y/4);
           bar(31*x,23*y,69*x,97*y/4);
           setcolor(forecolor);
           break;
     case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine  calls  graphs routine                                   */
/*************************************************************************/
void main()
{
  graph();
}
```

```
/***********************************************************************/
/* This routine gives an example of preorder traversal.               */
/***********************************************************************/
 void graph(void)
 {
  /*********************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-4");
  /*********************************************************************/
  outtextxy(2*x,2*y,"For simplicity we will identify the nodes with letters instead of
                    operations");
  outtextxy(2*x,3*y,"or operands.");
  outtextxy(2*x,4*y,"As we visit each vertex we will also keep the preorder list for
                    you to follow.");
  /*********************************************************************/
  pieslice(27*x,5*y,0,359,2);   /* H */
  pieslice(17*x,7*y,0,359,2);   /* F */
  pieslice(37*x,7*y,0,359,2);   /* N */
  moveto(17*x,7*y); lineto(27*x,5*y); lineto(37*x,7*y);
  outtextxy(27*x,9*y/2,"H");
  outtextxy(15*x,7*y,"F");
  outtextxy(39*x,7*y,"N");
  pieslice(12*x,9*y,0,359,2);   /* D */
  pieslice(22*x,9*y,0,359,2);   /* G */
  pieslice(32*x,9*y,0,359,2);   /* L */
  pieslice(42*x,9*y,0,359,2);   /* O */
  moveto(12*x,9*y); lineto(17*x,7*y); lineto(22*x,9*y);
  moveto(32*x,9*y); lineto(37*x,7*y); lineto(42*x,9*y);
  outtextxy(10*x,9*y,"D");
  outtextxy(22*x,19*y/2,"G");
  outtextxy(30*x,9*y,"L");
  outtextxy(42*x,19*y/2,"O");
  pieslice(7*x,11*y,0,359,2);    /* B */
pieslice(17*x,11*y,0,359,2);   /* E */
pieslice(27*x,11*y,0,359,2);   /* J */
pieslice(37*x,11*y,0,359,2);   /* M */
moveto(7*x,11*y); lineto(12*x,9*y); lineto(17*x,11*y);
moveto(27*x,11*y); lineto(32*x,9*y); lineto(37*x,11*y);
outtextxy(5*x,11*y,"B");
outtextxy(17*x,23*y/2,"E");
outtextxy(24*x,11*y,"J");
outtextxy(37*x,23*y/2,"M");
pieslice(2*x,13*y,0,359,2);   /* A */
pieslice(12*x,13*y,0,359,2);   /* C */
pieslice(22*x,13*y,0,359,2);   /* I */
pieslice(32*x,13*y,0,359,2);   /* K */
moveto(2*x,13*y); lineto(7*x,11*y); lineto(12*x,13*y);
moveto(22*x,13*y); lineto(27*x,11*y); lineto(32*x,13*y);
outtextxy(2*x,27*y/2,"A");
outtextxy(12*x,27*y/2,"C");
outtextxy(22*x,27*y/2,"I");
outtextxy(32*x,27*y/2,"K");
/*********************************************************************/
outtextxy(44*x,5*y,"Step by step Preorder Traversal");
moveto(43*x,11*y/2); lineto(89*x,11*y/2);
outtextxy(3*x,29*y/2,"Preorder  Listing");
moveto(2*x,15*y); lineto(40*x,15*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
setlinestyle(3,0,1);
/*********************************************************************/
outtextxy(44*x,6*y,"We will start out traversal by visiting");
outtextxy(44*x,7*y,"root H and put it in the preorder list");
outtextxy(3*x,16*y,"H");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
```

```
setcolor(forecolor);
moveto(44*x,15*y/2); lineto(89*x,15*y/2);
/*****************************************************************/
outtextxy(44*x,8*y,"Now we go to the left subtree of H and");
outtextxy(44*x,9*y,"start traversal again. This time we will");
outtextxy(44*x,10*y,"visit the root F and put it in the list.");
outtextxy(5*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,21*y/2); lineto(89*x,21*y/2);
/*****************************************************************/
outtextxy(44*x,11*y,"Next we will go to the left subtree of F");
outtextxy(44*x,12*y,"this time, and start traversal from this");
outtextxy(44*x,13*y,"point. We  will visit root D this time");
outtextxy(44*x,14*y,"and put it in the list.");
outtextxy(7*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,29*y/2); lineto(89*x,29*y/2);
/*****************************************************************/
outtextxy(44*x,15*y,"Now we will go to left subtree of D and");
outtextxy(44*x,16*y,"visit the root B and put it into list");
outtextxy(9*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,33*y/2); lineto(89*x,33*y/2);
/*****************************************************************/
outtextxy(44*x,17*y,"We now will go to the left subtree of");
outtextxy(44*x,18*y,"B and visit A and put it into the list.");
outtextxy(44*x,19*y,"As you see A does not have a left sub-");
outtextxy(44*x,20*y,"tree, so according to the algorithm we");
outtextxy(44*x,21*y,"will now visit the right subtree of");
outtextxy(44*x,22*y,"B (which consist of just the vertex C)");
outtextxy(44*x,23*y,"and put C to the list");
outtextxy(11*x,16*y,"A");
outtextxy(13*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(44*x,6*y,"Consequently, we next begin the traver-");
outtextxy(44*x,7*y,"sal of the right subtree of D.As you see");
outtextxy(44*x,8*y,"this subtree consists only of the ver-");
outtextxy(44*x,9*y,"tex E. So we will visit E and put it");
outtextxy(44*x,10*y,"into our list.");
outtextxy(15*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,21*y/2); lineto(89*x,21*y/2);
/*****************************************************************/
outtextxy(44*x,11*y,"Next we will do the traversal of ");
outtextxy(44*x,12*y,"right subtree of root F, visit ver-");
outtextxy(44*x,13*y,"tex G and put it into the list.");
outtextxy(17*x,16*y,"G");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,27*y/2); lineto(89*x,27*y/2);
/*****************************************************************/
outtextxy(44*x,14*y,"As you notice we have completed ");
outtextxy(44*x,15*y,"the traversal of the left subtree");
outtextxy(44*x,16*y,"of the root H. So now we will start");
outtextxy(44*x,17*y,"traversal of right subtree of H.");
outtextxy(44*x,18*y,"to do that we will visit root N and");
outtextxy(44*x,19*y,"go to the left subtree of N and ");
outtextxy(44*x,20*y,"begin another preorder traversal.");
outtextxy(19*x,16*y,"N");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
outtextxy(44*x,6*y,"We now visit root L and put it into");
outtextxy(44*x,7*y,"the list, and then go to the left ");
outtextxy(44*x,8*y,"subtree of L to begin another tra-");
outtextxy(44*x,9*y,"versal.");
outtextxy(21*x,16*y,"L");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,19*y/2); lineto(89*x,19*y/2);
/********************************************************************/
outtextxy(44*x,10*y,"This time we will visit root J.");
outtextxy(44*x,11*y,"We put it into the list, then we");
outtextxy(44*x,12*y,"go to the left subtree of J for ");
outtextxy(44*x,13*y,"another traversal.");
outtextxy(23*x,16*y,"J");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,27*y/2); lineto(89*x,27*y/2);
/********************************************************************/
outtextxy(44*x,14*y,"Left subtree of J consists of just");
outtextxy(44*x,15*y,"one vertex, namely vertex I. We vi-");
outtextxy(44*x,16*y,"sit this vertex and put it to the ");
outtextxy(44*x,17*y,"list. As you see I does not have");
outtextxy(44*x,18*y,"left subtree. So we go to right");
outtextxy(44*x,19*y,"subtree of J for another traversal");
outtextxy(25*x,16*y,"I");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,39*y/2); lineto(89*x,39*y/2);
/*******************************************************************/
outtextxy(44*x,20*y,"Right subtree of J consists of just");
outtextxy(44*x,21*y,"the vertex K. We visit this vertex");
outtextxy(44*x,22*y,"and put it into the list.");
outtextxy(27*x,16*y,"K");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"As you see we have completed the");
outtextxy(44*x,7*y,"traversal of the left subtree of");
outtextxy(44*x,8*y,"the root L. So now we will start");
outtextxy(44*x,9*y,"traversal of right subtree of L.");
outtextxy(44*x,10*y,"to do that we will visit M which");
outtextxy(44*x,11*y,"is the only vertex at the right");
outtextxy(44*x,12*y,"subtree of root L.");
outtextxy(29*x,16*y,"M");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,25*y/2); lineto(89*x,25*y/2);
/*******************************************************************/
outtextxy(44*x,13*y,"This last visit completed the ");
outtextxy(44*x,14*y,"traversal of the left subtree of");
outtextxy(44*x,15*y,"the root N. So now we will go to ");
outtextxy(44*x,16*y,"the right subtree of N and start.");
outtextxy(44*x,17*y,"another traversal. This traversal");
outtextxy(44*x,18*y,"consists only of visiting vertex O");
outtextxy(44*x,19*y,"and so completes the preorder tra-");
outtextxy(44*x,20*y,"versal of the entire binary tree.");
outtextxy(31*x,16*y,"O");
/**********************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM   : exb5.c
  AUTHOR     : Atilla BAKAN
  DATE       : Apr. 16, 1990
  REVISED    : Apr. 17, 1990

  DESCRIPTION : Fifth example about binary trees. It gives an example of
                preorder traversal.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
             C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph         (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code  */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/***********************************************************************/
/* This fuction initializes the necessary graphical routines           */
/***********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/***********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL); /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
   }
    switch (ch)          {
     case 'y': closegraph();
           videoinit();
           exit(0);
           break;
     case 'Y': closegraph();
           videoinit();
           exit(0);
           break;
     case 'n': setcolor(backcolor);
           bar(4*x/3,23*y,30*x,97*y/4);
           bar(31*x,23*y,69*x,97*y/4);
           setcolor(forecolor);
           break;
     case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
   settextstyle(0,0,0);
   setlinestyle(0,4,3);
   settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine  calls graph  routine                                    */
/*************************************************************************/
void main()
{
   graph();
}
```

```
/***********************************************************************/
/* This routine gives examples of trees and some graphs that are not trees.         */
/***********************************************************************/
 void graph(void)
 {
  /*********************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-5");
  /*********************************************************************/
  outtextxy(2*x,2*y,"For simplicity we will identify the nodes with letters instead of
                              operations");
  outtextxy(2*x,3*y,"or operands as we did in previous example.");
  outtextxy(2*x,4*y,"Again we visit each vertex we will also keep the preorder
                              list.");
  /*********************************************************************/
  pieslice(27*x,5*y,0,359,2);   /* G */
  pieslice(17*x,7*y,0,359,2);   /* D */
  pieslice(37*x,7*y,0,359,2);   /* H */
  moveto(17*x,7*y);  lineto(27*x,5*y);  lineto(37*x,7*y);
  outtextxy(27*x,9*y/2,"G");
  outtextxy(15*x,7*y,"D");
  outtextxy(39*x,7*y,"H");
  pieslice(12*x,9*y,0,359,2);   /* C */
  pieslice(22*x,9*y,0,359,2);   /* E */
  moveto(12*x,9*y);  lineto(17*x,7*y);  lineto(22*x,9*y);
  outtextxy(10*x,9*y,"C");
  outtextxy(22*x,19*y/2,"E");
  pieslice(7*x,11*y,0,359,2);    /* A */
  pieslice(18*x,11*y,0,359,2);   /* F */
  moveto(7*x,11*y);  lineto(12*x,9*y);
  moveto(22*x,9*y);  lineto(18*x,11*y);
  outtextxy(5*x,11*y,"A");
  outtextxy(18*x,23*y/2,"F");
```

```
pieslice(12*x,13*y,0,359,2);   /* B */
moveto(7*x,11*y); lineto(12*x,13*y);
outtextxy(12*x,27*y/2,"B");
/*****************************************************************/
outtextxy(44*x,5*y,"Step by step Preorder Traversal");
moveto(43*x,11*y/2); lineto(89*x,11*y/2);
outtextxy(3*x,29*y/2,"Preorder  Listing");
moveto(2*x,15*y); lineto(40*x,15*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
setlinestyle(3,0,1);
/*****************************************************************/
outtextxy(44*x,6*y,"We will start out traversal by visiting");
outtextxy(44*x,7*y,"root G and put it in the preorder list");
outtextxy(3*x,16*y,"G");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,15*y/2); lineto(89*x,15*y/2);
/*****************************************************************/
outtextxy(44*x,8*y,"Now we go to the left subtree of G and");
outtextxy(44*x,9*y,"start traversal again. This time we will");
outtextxy(44*x,10*y,"visit the root D and put it in the list.");
outtextxy(5*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,21*y/2); lineto(89*x,21*y/2);
/*****************************************************************/
outtextxy(44*x,11*y,"Next we will go to the left subtree of D");
outtextxy(44*x,12*y,"this time, and start traversal from this");
outtextxy(44*x,13*y,"point. We  will visit root C this time");
outtextxy(44*x,14*y,"and put it in the list.");
outtextxy(7*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,29*y/2); lineto(89*x,29*y/2);
/*****************************************************************/
outtextxy(44*x,15*y,"Next we will go to the left subtree of D");
outtextxy(44*x,16*y,"this time, and start traversal from this");
outtextxy(44*x,17*y,"point and visit root A this time and put");
outtextxy(44*x,18*y,"and put it in the list.");
outtextxy(9*x,16*y,"A");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,37*y/2); lineto(89*x,37*y/2);
/*****************************************************************/
outtextxy(44*x,19*y,"As you see A does not have left subtree.");
outtextxy(44*x,20*y,"So we then go to right subtree of A and");
outtextxy(44*x,21*y,"start traversal from there.We will visit");
outtextxy(44*x,22*y,"vertex B at the right subtree of A, and");
outtextxy(44*x,23*y,"put it into the list.");
outtextxy(11*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(44*x,6*y,"Since C does not have right subtree we");
outtextxy(44*x,7*y,"have completed traversing left subtree");
outtextxy(44*x,8*y,"of the root D. So we will go to the ");
outtextxy(44*x,9*y,"right subtree of D and start another ");
outtextxy(44*x,10*y,"preorder traversal. This time we visit");
outtextxy(44*x,11*y,"root E, and put it into the list.");
outtextxy(13*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,23*y/2); lineto(89*x,23*y/2);
/*******************************************************************/
outtextxy(44*x,12*y,"Then we go to the left subtree of E and");
outtextxy(44*x,13*y,"start another traversal again.Since this");
outtextxy(44*x,14*y,"subtree consists only of the vertex F");
outtextxy(44*x,15*y,"we will visit F and put it into list.");
outtextxy(15*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,31*y/2); lineto(89*x,31*y/2);
/*******************************************************************/
outtextxy(44*x,16*y,"Since neither F nor E has right sub-");
outtextxy(44*x,17*y,"tree we have completed traversal of");
outtextxy(44*x,18*y,"the left subtree of root G. Conse-");
outtextxy(44*x,19*y,"quently we go to the right subtree");
outtextxy(44*x,20*y,"of G where there is only the vertex");
outtextxy(44*x,21*y,"H. Visiting this vertex will complete");
outtextxy(44*x,22*y,"the preorder traversal of the entire");
outtextxy(44*x,23*y,"binary tree.");
outtextxy(17*x,16*y,"H");
/*******************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM   : exb6.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 16, 1990
   REVISED    : Apr. 17, 1990

   DESCRIPTION :Sixth example about binary trees. It gives an example of Polish
                 notation.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C  compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                       /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph         (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code  */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/**********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
           videoinit();
           exit(0);
           break;
     case 'Y': closegraph();
           videoinit();
           exit(0);
           break;
     case 'n': setcolor(backcolor);
           bar(4*x/3,23*y,30*x,97*y/4);
           bar(31*x,23*y,69*x,97*y/4);
           setcolor(forecolor);
           break;
     case 'N': setcolor(backcolor);
```

```
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine  calls graph  routine                                    */
/*************************************************************************/
void main()
{
  graph();
}
```

```
/**********************************************************************/
/* This routine gives an example of Polish notation.                 */
/**********************************************************************/
 void graph(void)
 {
  /*******************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-6");
  /*******************************************************************/
  outtextxy(2*x,2*y,"This time we will show you how to obtain Polish  notation from
                     the following");
  outtextxy(2*x,3*y,"expression  (((6  - 3) * 2) + 7) / (((5 - 1) * 4) + 8). While we
                     are doing");
  outtextxy(2*x,4*y,"this we won't tell you the details of our implementation, but we
                     will let");
  outtextxy(2*x,5*y,"you think about between each step. Later we'll try to do the re-
                     verse of this");
  outtextxy(2*x,6*y,"operation, that is obtaining the actual expression from the Pol-
                     ish notation.");
  /*******************************************************************/
  pieslice(27*x,8*y,0,359,2);   /* / */
  pieslice(17*x,10*y,0,359,2);  /* + */
  pieslice(37*x,10*y,0,359,2);  /* + */
  moveto(17*x,10*y); lineto(27*x,8*y); lineto(37*x,10*y);
  outtextxy(27*x,15*y/2,"/");
  outtextxy(15*x,10*y,"+");
  outtextxy(39*x,10*y,"+");
  pieslice(12*x,12*y,0,359,2);  /* * */
  pieslice(22*x,12*y,0,359,2);  /* 7 */
  pieslice(32*x,12*y,0,359,2);  /* * */
  pieslice(42*x,12*y,0,359,2);  /* 8 */
  moveto(12*x,12*y); lineto(17*x,10*y); lineto(22*x,12*y);
  moveto(32*x,12*y); lineto(37*x,10*y); lineto(42*x,12*y);
```

```
outtextxy(10*x,12*y,"*");
outtextxy(22*x,25*y/2,"7");
outtextxy(30*x,12*y,"*");
outtextxy(42*x,25*y/2,"8");
pieslice(7*x,14*y,0,359,2);    /* - */
pieslice(17*x,14*y,0,359,2);   /* 2 */
pieslice(27*x,14*y,0,359,2);   /* - */
pieslice(37*x,14*y,0,359,2);   /* 4 */
moveto(7*x,14*y); lineto(12*x,12*y); lineto(17*x,14*y);
moveto(27*x,14*y); lineto(32*x,12*y); lineto(37*x,14*y);
outtextxy(5*x,14*y,"-");
outtextxy(17*x,29*y/2,"2");
outtextxy(24*x,14*y,"-");
outtextxy(37*x,29*y/2,"4");
pieslice(2*x,16*y,0,359,2);    /* 6 */
pieslice(12*x,16*y,0,359,2);   /* 3 */
pieslice(22*x,16*y,0,359,2);   /* 5 */
pieslice(32*x,16*y,0,359,2);   /* 1 */
moveto(2*x,16*y); lineto(7*x,14*y); lineto(12*x,16*y);
moveto(22*x,16*y); lineto(27*x,14*y); lineto(32*x,16*y);
outtextxy(2*x,33*y/2,"6");
outtextxy(12*x,33*y/2,"3");
outtextxy(22*x,33*y/2,"5");
outtextxy(32*x,33*y/2,"1");
/*********************************************************************/
outtextxy(3*x,39*y/2,"Polish notation");
moveto(2*x,20*y); lineto(38*x,20*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,4*y/3,89*x,13*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
Pause(30*x,24*y);
outtextxy(3*x,21*y,"/");
setcolor(backcolor);
```

```
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
Pause(30*x,24*y);
outtextxy(5*x,21*y,"+");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
Pause(30*x,24*y);
outtextxy(7*x,21*y,"*");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
Pause(30*x,24*y);
outtextxy(9*x,21*y,"-");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
Pause(30*x,24*y);
outtextxy(11*x,21*y,"6");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
Pause(30*x,24*y);
outtextxy(13*x,21*y,"3");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
Pause(30*x,24*y);
outtextxy(15*x,21*y,"2");
setcolor(backcolor);
```

```
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
Pause(30*x,24*y);
outtextxy(17*x,21*y,"7");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
Pause(30*x,24*y);
outtextxy(19*x,21*y,"+");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
Pause(30*x,24*y);
outtextxy(21*x,21*y,"*");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
Pause(30*x,24*y);
outtextxy(23*x,21*y,"-");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
Pause(30*x,24*y);
outtextxy(25*x,21*y,"5");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
Pause(30*x,24*y);
outtextxy(27*x,21*y,"1");
setcolor(backcolor);
```

```
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
Pause(30*x,24*y);
outtextxy(29*x,21*y,"4");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
Pause(30*x,24*y);
outtextxy(31*x,21*y,"8");
/*******************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x/2,17*y,50*x,23*y);
setcolor(forecolor);
outtextxy(3*x,37*y/2,"Evolution of the expression");
moveto(2*x,19*y); lineto(50*x,19*y);
outtextxy(44*x,17*y/4,"Explanations for each step");
moveto(43*x,19*y/4); lineto(89*x,19*y/4);
/*******************************************************************/
outtextxy(2*x,2*y,"This time we will show you how to obtain our original expres-
                  sion back");
outtextxy(2*x,3*y,"To do this we will again apply the rule step by step for you to
                  follow.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"Start scanning from left to right.");
outtextxy(44*x,7*y,"Scan until you reach two numbers ");
outtextxy(44*x.8*y,"following an operation. (i.e. - 6 3)");
outtextxy(44*x,9*y,"Let T = -, a = 6, b = 3, by the rule");
outtextxy(44*x,10*y,"change T a b (i.e. - 6 3) with a T b.");
outtextxy(44*x,11*y,"This will turn  - 6 3 into (6 - 3)");
outtextxy(44*x,12*y,"Replace this expression in the Polish");
outtextxy(44*x,13*y,"notation with the old one.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x,39*y/2,52*x,23*y);
bar(43*x,5*y,179*x/2,18*y);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(3*x,20*y,"/ + * (6 - 3) 2 7 + * - 5 1 4 8");
/*******************************************************************/
outtextxy(44*x,6*y,"Once again scan from left to right");
outtextxy(44*x,7*y,"until you reach two numbers follow-");
outtextxy(44*x,8*y,"ing an operation. This time T = *, ");
outtextxy(44*x,9*y,"a = (6 - 3) (because result of this");
outtextxy(44*x,10*y,"expression is another number) and");
outtextxy(44*x,11*y,"b = 2; apply the rule and change");
outtextxy(44*x,12*y,"T a b (i.e. (* (6 - 3) 2 ) with ");
outtextxy(44*x,13*y,"a T b. Consequently we will have");
outtextxy(44*x,14*y,"        ((6 -3) * 2)        ");
outtextxy(44*x,15*y,"Replace this expression in the Polish");
outtextxy(44*x,16*y,"notation with the old one.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x,39*y/2,52*x,23*y);
bar(43*x,5*y,179*x/2,18*y);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(3*x,20*y,"/ + ((6 - 3) * 2) 7 + * - 5 1 4 8");
/*******************************************************************/
outtextxy(44*x,6*y,"Again scan from left to right until");
outtextxy(44*x,7*y,"you reach two numbers following an");
outtextxy(44*x,8*y,"operation. This time T = +, a= ((6 - 3)");
outtextxy(44*x,9*y,"* 2) (same reasoning that we did for ");
outtextxy(44*x,10*y,"(6 - 3). ) and b = 7; apply the rule");
outtextxy(44*x,11*y,"and change T a b (i.e + ((6 - 3) * 2))");
outtextxy(44*x,12*y,"with a T b. Consequently we will have");
outtextxy(44*x,13*y,"      (((6 -3) * 2) + 7)        ");
outtextxy(44*x,14*y,"Replace this expression in the Polish");
outtextxy(44*x,15*y,"notation with the old one.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x,39*y/2,52*x,23*y);
bar(43*x,5*y,179*x/2,18*y);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(3*x,20*y,"/ (((6 - 3) * 2) + 7) + * - 5 1 4 8");
/**********************************************************************/
outtextxy(44*x,6*y,"Again scan from left to right until");
outtextxy(44*x,7*y,"you reach two numbers following an");
outtextxy(44*x,8*y,"operation. This time T = -, a= 5 and");
outtextxy(44*x,9*y,"b = 1, (you realized that we skipped ");
outtextxy(44*x,10*y,"(((6 - 3) * 2) + 7); because it is by");
outtextxy(44*x,11*y,"itself a number and is followed by +");
outtextxy(44*x,12*y,"operation). Now we will apply the rule");
outtextxy(44*x,13*y,"and change T a b (i.e - 5 1) with a T b");
outtextxy(44*x,14*y,"Consequently we will have (5 - 1) and we");
outtextxy(44*x,15*y,"replace this expression in the Polish");
outtextxy(44*x,16*y,"notation with the old one.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x,39*y/2,52*x,23*y);
bar(43*x,5*y,179*x/2,18*y);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(3*x,20*y,"/ (((6 - 3) * 2) + 7) + * (5 - 1) 4 8");
/**********************************************************************/
outtextxy(44*x,6*y,"Scan from left to right until you");
outtextxy(44*x,7*y,"reach two numbers following an oper-");
outtextxy(44*x,8*y,"ation. This time T = *, a= (5 - 1) and");
outtextxy(44*x,9*y,"b = 4.Now we will apply the rule and");
outtextxy(44*x,10*y,"change T a b (i.e - 5 1) with a T b");
outtextxy(44*x,11*y,"Consequently we will have ((5 - 1) * 4)");
outtextxy(44*x,12*y,"and we replace this expression in the");
outtextxy(44*x,13*y,"Polish notation with the old one.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x,39*y/2,52*x,23*y);
bar(43*x,5*y,179*x/2,18*y);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(3*x,20*y,"/ (((6 - 3) * 2) + 7) + ((5 - 1) * 4) 8");
/*****************************************************************************/
outtextxy(44*x,6*y,"You know our reasonings, from this time");
outtextxy(44*x,7*y,"we won't make explanations but just tell");
outtextxy(44*x,8*y,"what T , a and b are.So, this time T = *");
outtextxy(44*x,9*y,"a= ((5 - 1) * 4) and b = 8.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x,39*y/2,52*x,23*y);
bar(43*x,5*y,179*x/2,18*y);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(3*x,20*y,"/ (((6 - 3) * 2) + 7) (((5 - 1) * 4) * 8)");
/*****************************************************************************/
outtextxy(44*x,6*y,"This time T = / a = (((6 - 3) * 2) + 7)");
outtextxy(44*x,7*y,"and b = (((5 -1) * 4) * 8).");
Pause(30*x,24*y);
setcolor(backcolor);
bar(3*x,39*y/2,52*x,23*y);
bar(43*x,5*y,179*x/2,18*y);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(3*x.20*y,"(((6 - 3) * 2) + 7) / (((5 - 1) * 4) * 8)");
/*****************************************************************************/
outtextxy(44*x,6*y,"As you see since there is no operation ");
outtextxy(44*x,7*y,"which is followed by two numbers. This");
outtextxy(44*x,8*y,"means we are done. Actually when you ");
outtextxy(44*x,9*y,"examine the resultant expression you will");
outtextxy(44*x,10*y,"see that it is what we started with.");
Pause(30*x,24*y);
/**************************************************************************/
closegraph();
videoinit();
}
```

```
/* PROGRAM   : exb7.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 16, 1990
   REVISED   : Apr. 17, 1990

   DESCRIPTION : Seventh example about binary trees. It gives an example
                 of postorder traversal.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)           /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name          name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name          name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph         (void);

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/**********************************************************************/
/* This function is used for including drivers to the executable code */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/**************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kbiist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
   }
   switch (ch)         {
    case 'y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
```

```
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/************************************************************************/
/* This function sets the text default values                          */
/************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/************************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/************************************************************************/
/* main routine calls graph  routine                                   */
/************************************************************************/
void main()
{
  graph();
}
```

```
/***************************************************************************/
/* This routine gives an example of postorder traversal.                  */
/***************************************************************************/
 void graph(void)
 {
  /************************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-7");
  /************************************************************************/
  outtextxy(2*x,2*y,"To make you able to compare the result we will use the result,
                     we will again");
  outtextxy(2*x,3*y,"use the same expression and apply the algorithm on them. And
                     as we did before");
  outtextxy(2*x,4*y,"as we visit each vertex we will keep the postorder list for you
                     to follow.");
  /************************************************************************/
  pieslice(27*x,5*y,0,359,2);   /* H */
  pieslice(17*x,7*y,0,359,2);   /* F */
  pieslice(37*x,7*y,0,359,2);   /* N */
  moveto(17*x,7*y); lineto(27*x,5*y); lineto(37*x,7*y);
  outtextxy(27*x,9*y/2,"H");
  outtextxy(15*x,7*y,"F");
  outtextxy(39*x,7*y,"N");
  pieslice(12*x,9*y,0,359,2);   /* D */
  pieslice(22*x,9*y,0,359,2);   /* G */
  pieslice(32*x,9*y,0,359,2);   /* L */
  pieslice(42*x,9*y,0,359,2);   /* O */
  moveto(12*x,9*y); lineto(17*x,7*y); lineto(22*x,9*y);
  moveto(32*x,9*y); lineto(37*x,7*y); lineto(42*x,9*y);
  outtextxy(10*x,9*y,"D");
  outtextxy(22*x,19*y/2,"G");
  outtextxy(30*x,9*y,"L");
  outtextxy(42*x,19*y/2,"O");
```

```
pieslice(7*x,11*y,0,359,2);    /* B */
pieslice(17*x,11*y,0,359,2);   /* E */
pieslice(27*x,11*y,0,359,2);   /* J */
pieslice(37*x,11*y,0,359,2);   /* M */
moveto(7*x,11*y); lineto(12*x,9*y); lineto(17*x,11*y);
moveto(27*x,11*y); lineto(32*x,9*y); lineto(37*x,11*y);
outtextxy(5*x,11*y,"B");
outtextxy(17*x,23*y/2,"E"`
outtextxy(24*x,11*y,"J");
outtextxy(37*x,23*y/2,"M");
pieslice(2*x,13*y,0,359,2);    /* A */
pieslice(12*x,13*y,0,359,2);   /* C */
pieslice(22*x,13*y,0,359,2);   /* I */
pieslice(32*x,13*y,0,359,2);   /* K */
moveto(2*x,13*y); lineto(7*x,11*y); lineto(12*x,13*y);
moveto(22*x,13*y); lineto(27*x,11*y); lineto(32*x,13*y);
outtextxy(2*x,27*y/2,"A");
outtextxy(12*x,27*y/2,"C");
outtextxy(22*x,27*y/2,"I");
outtextxy(32*x,27*y/2,"K");
/*****************************************************************/
outtextxy(44*x,5*y,"Step by step Postorder Traversal");
moveto(43*x,11*y/2); lineto(89*x,11*y/2);
outtextxy(3*x,29*y/2,"Postorder  Listing");
moveto(2*x,15*y); lineto(40*x,15*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
setlinestyle(3,0,1);
/*****************************************************************/
outtextxy(44*x,6*y,"We will start out traversal by visiting");
outtextxy(44*x,7*y,"left subtree (roote by F), since there");
outtextxy(44*x,8*y,"is one we will apply postorder traversal");
outtextxy(44*x,9*y,"Once again since F has a left subtree");
outtextxy(44*x,10*y,"rooted by D, we will apply postorder ");
outtextxy(44*x,11*y,"traversal and go to B. B too, has left");
outtextxy(44*x,12*y,"subtree, so we will apply the algorithm");
outtextxy(44*x,13*y,"once again. Since A is a terminal node");
outtextxy(44*x,14*y,"it does not have neither left nor right");
outtextxy(44*x,15*y,"child (in other word it is the root of");
outtextxy(44*x,16*y,"its own) we will visit A");
outtextxy(3*x,16*y,"A");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,33*y/2); lineto(89*x,33*y/2);
/********************************************************************/
outtextxy(44*x,17*y,"Now we go to the right subtree of B and");
outtextxy(44*x,18*y,"start traversal again.This time we will");
outtextxy(44*x,19*y,"go to C. C does not have children we");
outtextxy(44*x,20*y,"will visit C.");
outtextxy(5*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,41*y/2); lineto(89*x,41*y/2);
/********************************************************************/
outtextxy(44*x,21*y,"Next we will go to the root B and since");
outtextxy(44*x,22*y,"visited its left & right child we will");
outtextxy(44*x,23*y,"visit B");
outtextxy(7*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
outtextxy(44*x,6*y,"We completed traversal of left subtree");
outtextxy(44*x,7*y,"of D we will got its right subtree, E");
outtextxy(44*x,8*y,"Since E is a terminal node we'll visit E");
outtextxy(9*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,17*y/2); lineto(89*x,17*y/2);
/*******************************************************************/
outtextxy(44*x,9*y,"We now will visit root D since we comp-");
outtextxy(44*x,10*y,"leted traversal of its subtrees.");
outtextxy(11*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,21*y/2); lineto(89*x,21*y/2);
/*******************************************************************/
outtextxy(44*x,11*y,"Consequently, we next begin the traver-");
outtextxy(44*x,12*y,"sal of the right subtree of F.As you see");
outtextxy(44*x,13*y,"this subtree consists only of the ver-");
outtextxy(44*x,14*y,"tex G. So we will visit G and put it");
outtextxy(44*x,15*y,"into our list.");
outtextxy(13*x,16*y,"G");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,31*y/2); lineto(89*x,31*y/2);
/*******************************************************************/
outtextxy(44*x,16*y,"We now will visit root F since we comp-");
outtextxy(44*x,17*y,"leted traversal of its  subtrees.");
outtextxy(15*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
```

```
setcolor(forecolor);
/***********************************************************************/
outtextxy(44*x,6*y,"As you notice we have completed ");
outtextxy(44*x,7*y,"the traversal of the left subtree");
outtextxy(44*x,8*y,"of the root H. So now we will start");
outtextxy(44*x,9*y,"traversal of right subtree of H.");
outtextxy(44*x,10*y,"to do that we will go to root N and");
outtextxy(44*x,11*y,"go to the left subtree of N and ");
outtextxy(44*x,12*y,"begin another postorder traversal.");
outtextxy(44*x,13*y,"this traversal will take us down");
outtextxy(44*x,14*y,"to the terminal node I. So we will");
outtextxy(44*x,15*y,"visit I.");
outtextxy(17*x,16*y,"I");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,31*y/2); lineto(89*x,31*y/2);
/***********************************************************************/
outtextxy(44*x,16*y,"We now go to the right subtree of");
outtextxy(44*x,17*y,"root J. Right subtree of J contains");
outtextxy(44*x,18*y,"only the vertex K, so we'll visit K");
outtextxy(19*x,16*y,"K");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,37*y/2); lineto(89*x,37*y/2);
/***********************************************************************/
outtextxy(44*x,19*y,"We now will visit root J since we comp-");
outtextxy(44*x,20*y,"leted traversal of its subtrees.");
outtextxy(21*x,16*y,"J");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
```

```
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"Consequently, we next begin the traver-");
outtextxy(44*x,7*y,"sal of the right subtree of L.As you see");
outtextxy(44*x,8*y,"this subtree consists only of the ver-");
outtextxy(44*x,9*y,"tex M. So we will visit M ");
outtextxy(23*x,16*y,"M");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,19*y/2); lineto(89*x,19*y/2);
/*******************************************************************/
outtextxy(44*x,10*y,"We now will visit root L since we comp-");
outtextxy(44*x,11*y,"leted traversal of its subtrees.");
outtextxy(25*x,16*y,"L");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,23*y/2); lineto(89*x,23*y/2);
/*******************************************************************/
outtextxy(44*x,12*y,"We next begin the traversal of right");
outtextxy(44*x,13*y,"subtree of root N. As you see this ");
outtextxy(44*x,14*y,"subtree consists only of the vertex");
outtextxy(44*x,15*y,"O. So we will visit O and put it");
outtextxy(44*x,16*y,"into our list.");
outtextxy(27*x,16*y,"O");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,33*y/2); lineto(89*x,33*y/2);
/*******************************************************************/
outtextxy(44*x,17*y,"We now will visit root N since we comp-");
outtextxy(44*x,18*y,"leted traversal of its  subtrees.");
outtextxy(29*x,16*y,"N");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,37*y/2); lineto(89*x,37*y/2);
/*******************************************************************/
outtextxy(44*x,19*y,"This last visit completed the traversals");
outtextxy(44*x,20*y,"of the left and right subtrees of root");
outtextxy(44*x,21*y,"H. So now we will finally visit the root ");
outtextxy(44*x,22*y,"H. This will complete our postorder tra-");
outtextxy(44*x,23*y,"versal of the binary tree.");
outtextxy(31*x,16*y,"H");
/*******************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM   : exb8.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 16, 1990
   REVISED    : Apr. 17, 1990

   DESCRIPTION : Eighth example about binary trees. ,It gives an example of
                 postorder traversal.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>               /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph         (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code  */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**********************************************************************/
/* This fuction initializes the necessary graphical routines         */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
       }
  forecolor = WHITE;
  }
```

```
/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
```

```
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                           */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen               */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }
/*************************************************************************/
/* main routine calls graph routine                                      */
/*************************************************************************/
void main()
{
  graph();
}
```

```
/***********************************************************************/
/* This routine gives  an example of postorder traversal.              */
/***********************************************************************/
 void graph(void)
 {
  /*********************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-8");
  /*********************************************************************/
  outtextxy(2*x,2*y,"To make you able to compare the result we will use the result,
                    we will again");
  outtcxtxy(2*x,3*y,"use the same expression and apply the algorithm on them. And
                     as we did before");
  outtextxy(2*x,4*y,"as we visit each vertex we will keep the postorder list for you
                     to follow.");
  /*********************************************************************/
  pieslice(27*x,5*y,0,359,2);   /* G */
  pieslice(17*x,7*y,0,359,2);   /* D */
  pieslice(37*x,7*y,0,359,2);   /* H */
  moveto(17*x,7*y); lineto(27*x,5*y); lineto(37*x,7*y);
  outtextxy(27*x,9*y/2,"G");
  outtextxy(15*x,7*y,"D");
  outtextxy(39*x,7*y,"H");
  pieslice(12*x,9*y,0,359,2);   /* C */
  pieslice(22*x,9*y,0,359,2);   /* E */
  moveto(12*x,9*y); lineto(17*x,7*y); lineto(22*x,9*y);
  outtextxy(10*x,9*y,"C");
  outtextxy(22*x,19*y/2,"E");
  pieslice(7*x,11*y,0,359,2);    /* A */
  pieslice(18*x,11*y,0,359,2);   /* F */
  moveto(7*x,11*y); lineto(12*x,9*y);
  moveto(22*x,9*y); lineto(18*x,11*y);
  outtextxy(5*x,11*y,"A");
outtextxy(18*x,23*y/2,"F");
pieslice(12*x,13*y,0,359,2);   /* B */
moveto(7*x,11*y); lineto(12*x,13*y);
outtextxy(12*x,27*y/2,"B");
/*****************************************************************/
outtextxy(44*x,5*y,"Step by step Postorder Traversal");
moveto(43*x,11*y/2); lineto(89*x,11*y/2);
outtextxy(3*x,29*y/2,"Postorder  Listing");
moveto(2*x,15*y); lineto(40*x,15*y);
outtextxy(30*x,24*y,"PRESS ANY KEY TO CONTINUE...");
while(kbhit() ) getch();
getch();
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
setlinestyle(3,0,1);
/*****************************************************************/
outtextxy(44*x,6*y,"We will start out traversal by visiting");
outtextxy(44*x,7*y,"left subtree (roote by D), since there");
outtextxy(44*x,8*y,"is a one we will apply postorder traver-");
outtextxy(44*x,9*y,"sal.Once again since D has left subtree");
outtextxy(44*x,10*y,"rooted by C, we will apply postorder ");
outtextxy(44*x,11*y,"traversal and go to C. C too, has a left");
outtextxy(44*x,12*y,"subtree, so we will apply the algorithm");
outtextxy(44*x,13*y,"once again. We will go to A. A does not");
outtextxy(44*x,14*y,"left subtree but it has a right subtree");
outtextxy(44*x,15*y,"(contains only one vertex) B. So we will");
outtextxy(44*x,16*y,"go to B since it does not have any child");
outtextxy(44*x,17*y,"we will visit B.");
outtextxy(3*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,35*y/2); lineto(89*x,35*y/2);
/*****************************************************************/
```

```
outtextxy(44*x,18*y,"This will complete the travesals of A's ");
outtextxy(44*x,19*y,"subtrees. So, we will visit A");
outtextxy(5*x,16*y,"A");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(44*x,6*y,"This also completed the traversal of C's");
outtextxy(44*x,7*y,"left subtree.Since C does not have right");
outtextxy(44*x,8*y,"subtree we will visit root C this time");
outtextxy(44*x,9*y,"and put it in the list.");
outtextxy(7*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,19*y/2); lineto(89*x,19*y/2);
/*****************************************************************/
outtextxy(44*x,10*y,"Now we will go to the right subtree of D");
outtextxy(44*x,11*y,"this time, and start traversal from this");
outtextxy(44*x,12*y,"point and go to left subtree of E and");
outtextxy(44*x,13*y,"start postorder traversal again. We will");
outtextxy(44*x,14*y,"go to vertex F, since it is a terminal");
outtextxy(44*x,15*y,"vertex, we will visit it.");
outtextxy(9*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,31*y/2); lineto(89*x,31*y/2);
/*****************************************************************/
outtextxy(44*x,16*y,"This also completed the traversal of E's");
outtextxy(44*x,17*y,"left subtree.Since E does not have right");
outtextxy(44*x,18*y,"subtree we will visit root E this time");
outtextxy(44*x,19*y,"and put it in the list.");
outtextxy(11*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,39*y/2); lineto(89*x,39*y/2);
/*****************************************************************/
outtextxy(44*x,20*y,"We now will visit root D since we comp-");
outtextxy(44*x,21*y,"leted traversal of its subtrees.");
outtextxy(13*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(44*x,6*y,"Consequently, we next begin the traver-");
outtextxy(44*x,7*y,"sal of the right subtree of G.As you see");
outtextxy(44*x,8*y,"this subtree consists only of the ver-");
outtextxy(44*x,9*y,"tex H. So we will visit H and put it");
outtextxy(44*x,10*y,"into our list.");
outtextxy(15*x,16*y,"H");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,21*y/2); lineto(89*x,21*y/2);
/*****************************************************************/
outtextxy(44*x,11*y,"This last visit completed the traversal");
outtextxy(44*x,12*y,"of the left and right subtrees of root");
outtextxy(44*x,13*y,"G. Visiting this vertex will complete");
outtextxy(44*x,14*y,"the postorder traversal of the entire");
outtextxy(44*x,15*y,"binary tree.");
outtextxy(17*x,16*y,"G");
/*****************************************************************/
```

```
 Pause(30*x,24*y);
 closegraph();
 videoinit();
}
```

```
/* PROGRAM   : exb9.c
  AUTHOR     : Atilla BAKAN
  DATE       : Apr. 16, 1990
  REVISED    : Apr. 17, 1990

  DESCRIPTION : Ninth example about binary trees. It gives an exanple of
                inorder traversal.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
            C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk           find_t
  #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph        (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code .  */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/************************************************************************/
/* This fuction initializes the necessary graphical routines           */
/************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
   }
   switch (ch)         {
    case 'y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
```

```
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine  calls graph  routine                                    */
/*************************************************************************/
void main()
{
  graph();
}
```

```
/***********************************************************************/
/* This routine gives an example  of  inorder traversal.                */
/***********************************************************************/
 void graph(void)
 {
  /********************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-9");
  /********************************************************************/
  outtextxy(2*x,2*y,"To make you able to compare the result we will use the result,
                      we will again");
  outtextxy(2*x,3*y,"use the same expression and apply the algorithm on them. And
                      as we did before");
  outtextxy(2*x,4*y,"as we visit each vertex we will keep the inorder list for you to
                      follow.");
  /********************************************************************/
  pieslice(27*x,5*y,0,359,2);   /* H */
  pieslice(17*x,7*y,0,359,2);   /* F */
  pieslice(37*x,7*y,0,359,2);   /* N */
  moveto(17*x,7*y); lineto(27*x,5*y); lineto(37*x,7*y);
  outtextxy(27*x,9*y/2,"H");
  outtextxy(15*x,7*y,"F");
  outtextxy(39*x,7*y,"N");
  pieslice(12*x,9*y,0,359,2);   /* D */
  pieslice(22*x,9*y,0,359,2);   /* G */
  pieslice(32*x,9*y,0,359,2);   /* L */
  pieslice(42*x,9*y,0,359,2);   /* O */
  moveto(12*x,9*y); lineto(17*x,7*y); lineto(22*x,9*y);
  moveto(32*x,9*y); lineto(37*x,7*y); lineto(42*x,9*y);
  outtextxy(10*x,9*y,"D");
  outtextxy(22*x,19*y/2,"G");
  outtextxy(30*x,9*y,"L");
  outtextxy(42*x,19*y/2,"O");
```

```
pieslice(7*x,11*y,0,359,2);    /* B */
pieslice(17*x,11*y,0,359,2);   /* E */
pieslice(27*x,11*y,0,359.2);   /* J */
pieslice(37*x,11*y,0,359,2);   /* M */
moveto(7*x,11*y); lineto(12*x,9*y); lineto(17*x,11*y);
moveto(27*x,11*y); lineto(32*x,9*y); lineto(37*x,11*y);
outtextxy(5*x,11*y,"B");
outtextxy(17*x,23*y/2,"E");
outtextxy(24*x,11*y,"J");
outtextxy(37*x,23*y/2,"M");
pieslice(2*x,13*y,0,359,2);    /* A */
pieslice(12*x,13*y,0,359,2);   /* C */
pieslice(22*x,13*y,0,359,2);   /* I */
pieslice(32*x,13*y,0,359,2);   /* K */
moveto(2*x,13*y); lineto(7*x,11*y); lineto(12*x,13*y);
moveto(22*x.13*y); lineto(27*x,11*y); lineto(32*x,13*y);
outtextxy(2*x.27*y/2,"A");
outtextxy(12*x,27*y/2,"C");
outtextxy(22*x,27*y/2,"I");
outtextxy(32*x,27*y/2,"K");
/*********************************************************************/
outtextxy(44*x,5*y,"Step by step Inorder Traversal");
moveto(43*x,11*y/2); lineto(89*x,11*y/2);
outtextxy(3*x,29*y/2,"Inorder  Listing");
moveto(2*x,15*y); lineto(40*x,15*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
setlinestyle(3,0,1);
/*********************************************************************/
outtextxy(44*x,6*y,"We will start out traversal by visiting");
outtextxy(44*x,7*y,"left subtree (roote by F), since there");
outtextxy(44*x,8*y,"is one we will apply inorder traversal");
outtextxy(44*x,9*y,"Once again since F has a left subtree");
outtextxy(44*x,10*y,"rooted by D, we will apply inorder ");
outtextxy(44*x,11*y,"traversal and go to B. B too, has left");
outtextxy(44*x,12*y,"subtree, so we will apply the algorithm");
outtextxy(44*x,13*y,"once again. Since A is a terminal node");
outtextxy(44*x,14*y,"it does not have neither left nor right");
outtextxy(44*x,15*y,"child (in other word it is the root of");
outtextxy(44*x,16*y,"its own) we will visit A");
outtextxy(3*x,16*y,"A");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,33*y/2); lineto(89*x,33*y/2);
/******************************************************************/
outtextxy(44*x,17*y,"Now we will visit root B.");
outtextxy(5*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,35*y/2); lineto(89*x,35*y/2);
/******************************************************************/
outtextxy(44*x,18*y,"Next we will go to the right child of B");
outtextxy(44*x,19*y,"which is C and visit it.");
outtextxy(7*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,39*y/2); lineto(89*x,39*y/2);
/******************************************************************/
outtextxy(44*x,20*y,"We completed traversal of left subtree");
outtextxy(44*x,21*y,"of D so we will D now.");
outtextxy(9*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
```

```
setcolor(forecolor);
moveto(44*x,43*y/2); lineto(89*x,43*y/2);
/*******************************************************************/
outtextxy(44*x,22*y,"We now will visit right child of D");
outtextxy(11*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,45*y/2); lineto(89*x,45*y/2);
/*******************************************************************/
outtextxy(44*x,23*y,"Consequently, we next visit root F.");
outtextxy(13*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"We now will visit right subtree of  F");
outtextxy(44*x,7*y,"which is G.");
outtextxy(15*x,16*y,"G");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,15*y/2); lineto(89*x,15*y/2);
/*******************************************************************/
outtextxy(44*x,8*y,"As you notice we have completed ");
outtextxy(44*x,9*y,"the traversal of the left subtree");
outtextxy(44*x,10*y,"of the root H. So now we will visit");
outtextxy(44*x,11*y,"the root H according to the algorithm.");
outtextxy(17*x,16*y,"H");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
```

```
setcolor(forecolor);
moveto(44*x,23*y/2); lineto(89*x,23*y/2);
/*****************************************************************/
outtextxy(44*x,12*y,"We now go to the right subtree of");
outtextxy(44*x,13*y,"root H. By following the algorithm");
outtextxy(44*x,14*y,"we will go down to the terminal node");
outtextxy(44*x,15*y,"I. Since we cannot go any further we");
outtextxy(44*x,16*y,"will visit I.");
outtextxy(19*x,16*y,"I");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,33*y/2); lineto(89*x,33*y/2);
/*****************************************************************/
outtextxy(44*x,17*y,"We now will visit root J.");
outtextxy(21*x,16*y,"J");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,35*y/2); lineto(89*x,35*y/2);
/*****************************************************************/
outtextxy(44*x,18*y,"Consequently, we next begin the traver-");
outtextxy(44*x,19*y,"sal of the right subtree of J.As you see");
outtextxy(44*x,20*y,"this subtree consists only of the ver-");
outtextxy(44*x,21*y,"tex K. So we will visit K. ");
outtextxy(23*x,16*y,"K");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,43*y/2); lineto(89*x,43*y/2);
/*****************************************************************/
outtextxy(44*x,22*y,"We now will visit root L .");
outtextxy(25*x,16*y,"L");
```

```
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"We next begin the traversal of right");
outtextxy(44*x,7*y,"subtree of root L. As you see this ");
outtextxy(44*x,8*y,"subtree consists only of the vertex");
outtextxy(44*x,9*y,"M. So we will visit M and put it");
outtextxy(44*x,10*y,"into our list.");
outtextxy(27*x,16*y,"M");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,21*y/2); lineto(89*x,21*y/2);
/*******************************************************************/
outtextxy(44*x,11*y,"We now will visit root N.");
outtextxy(29*x,16*y,"N");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,23*y/2); lineto(89*x,23*y/2);
/*******************************************************************/
outtextxy(44*x,12*y,"According to the algorithm we now will");
outtextxy(44*x,13*y,"visit right subtree of root N, O. This");
outtextxy(44*x,14*y,"will complete our inorder traversal");
outtextxy(44*x,15*y,"of the binary tree.");
outtextxy(31*x,16*y,"O");
/*******************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM  : exb10.c
   AUTHOR     : Atilla BAKAN
   DATE         : Apr. 16, 1990
   REVISED    : Apr. 17, 1990

   DESCRIPTION : Tenth example about binary trees. It gives an example of
                         inorder traversal.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                                     C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                          /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                        /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)            /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                             /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions     */
static void graph        (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/***********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/***********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
   }
```

```
/***********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
            videoinit();
            exit(0);
            break;
     case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
     case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     case 'N': setcolor(backcolor);
```

```
                bar(4*x/3,23*y,30*x,97*y/4);
                bar(31*x,23*y,69*x,97*y/4);
                setcolor(forecolor);
                break;
        default : break;
        }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);      /* restore any hidden hot keys */
}
/**********************************************************************/
/* This function sets the text default values                        */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/**********************************************************************/
/* main routine  calls graph  routine                                 */
/**********************************************************************/
void main()
{
  graph();
}
```

```
/***********************************************************************/
/* This routine gives an example of inorder traversal.                 */
/***********************************************************************/
 void graph(void)
 {
  /********************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/8);
  outtextxy(38*x,y/2,"EXAMPLE 4-5-10");
  /********************************************************************/
  outtextxy(2*x,2*y,"To make you able to compare the result we will use the result,
                     we will again");
  outtextxy(2*x,3*y,"use the same expression and apply the algorithm on them. And
                     as we did before");
  outtextxy(2*x,4*y,"as we visit each vertex we will keep the inorder list for you to
                     follow.");
  /********************************************************************/
  pieslice(27*x,5*y,0,359,2);   /* G */
  pieslice(17*x,7*y,0,359,2);   /* D */
  pieslice(37*x,7*y,0,359,2);   /* H */
  moveto(17*x,7*y); lineto(27*x,5*y); lineto(37*x,7*y);
  outtextxy(27*x,9*y/2,"G");
  outtextxy(15*x,7*y,"D");
  outtextxy(39*x,7*y,"H");
  pieslice(12*x,9*y,0,359,2);   /* C */
  pieslice(22*x,9*y,0,359,2);   /* F */
  moveto(12*x,9*y); lineto(17*x,7*y); lineto(22*x,9*y);
  outtextxy(10*x,9*y,"C");
  outtextxy(22*x,19*y/2,"F");
  pieslice(7*x,11*y,0,359,2);    /* A */
  pieslice(18*x,11*y,0,359,2);   /* E */
  moveto(7*x,11*y); lineto(12*x,9*y);
  moveto(22*x,9*y); lineto(18*x,11*y);
  outtextxy(5*x,11*y,"A");
outtextxy(18*x,23*y/2,"E");
pieslice(12*x,13*y,0,359,2);   /* B */
moveto(7*x,11*y); lineto(12*x,13*y);
outtextxy(12*x,27*y/2,"B");
/**********************************************************************/
outtextxy(44*x,5*y,"Step by step Inorder Traversal");
moveto(43*x,11*y/2); lineto(89*x,11*y/2);
outtextxy(3*x,29*y/2,"Inorder  Listing");
moveto(2*x,15*y); lineto(40*x,15*y);
outtextxy(30*x,24*y,"PRESS ANY KEY TO CONTINUE...");
while(kbhit() ) getch();
getch();
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
setlinestyle(3,0,1);
/**********************************************************************/
outtextxy(44*x,6*y,"We will start out traversal by visiting");
outtextxy(44*x,7*y,"left subtree (roote by D), since there");
outtextxy(44*x,8*y,"is a one we will apply inorder traver-");
outtextxy(44*x,9*y,"sal.Once again since D has left subtree");
outtextxy(44*x,10*y,"rooted by C, we will apply inorder ");
outtextxy(44*x,11*y,"traversal and go to C. C too, has a left");
outtextxy(44*x,12*y,"subtree, so we will apply the algorithm");
outtextxy(44*x,13*y,"once again. We will go to A. A does not");
outtextxy(44*x,14*y,"left subtree so according to the algo-");
outtextxy(44*x,15*y,"rithm we will visit A.");
outtextxy(3*x,16*y,"A");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,31*y/2); lineto(89*x,31*y/2);
/**********************************************************************/
outtextxy(44*x,16*y,"Then we will visit right subtree B. ");
outtextxy(5*x,16*y,"B");
```

```
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,33*y/2); lineto(89*x,33*y/2);
/*******************************************************************/
outtextxy(44*x,17*y,"This also completed the traversal of C's");
outtextxy(44*x,18*y,"left subtree. So now we will visit C.");
outtextxy(7*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,37*y/2); lineto(89*x,37*y/2);
/*******************************************************************/
outtextxy(44*x,19*y,"Now we will visit  D.");
outtextxy(9*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
moveto(44*x,39*y/2); lineto(89*x,39*y/2);
/*******************************************************************/
outtextxy(44*x,20*y,"This time we will go to F and start");
outtextxy(44*x,21*y,"inorder traversal from F. We will go");
outtextxy(44*x,22*y,"to left subtree of F, consisting of");
outtextxy(44*x,23*y,"only vertex E, so we will visit E.");
outtextxy(11*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"We now will visit root F since we comp-");
outtextxy(44*x,7*y,"leted traversal of its left subtree.");
```

```
  outtextxy(13*x,16*y,"F");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
  setcolor(forecolor);
  moveto(44*x,15*y/2); lineto(89*x,15*y/2);
  /*******************************************************************/
  outtextxy(44*x,8*y,"Consequently, we next visit the root G.");
  outtextxy(15*x,16*y,"G");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
  setcolor(forecolor);
  moveto(44*x,17*y/2); lineto(89*x,17*y/2);
  /*******************************************************************/
  outtextxy(44*x,9*y,"Finally we will go the right subtree");
  outtextxy(44*x,10*y,"of the root G. We have only one vertex");
  outtextxy(44*x,11*y,"on this subtree and visiting this");
  outtextxy(44*x,12*y,"vertex will complete the inorder");
  outtextxy(44*x,13*y,"traversal of the entire binary tree.");
  outtextxy(17*x,16*y,"H");
  /*******************************************************************/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
}
```

```
/* PROGRAM    : q451.c
   AUTHOR     : Atilla BAKAN
   DATE       : Mar. 22, 1990
   REVISED    : Mar. 22, 1990

   DESCRIPTION : This program contains the first exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                  /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                   /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph     (void);
static void confirm_graph_exit (void);
static void Pause          (int i, int j);
static void register_drivers (void);
extern void settext        (void);

/* tutorial functions      */
static void exer           (void);
static void example        (void);
static void show_alg       (void);
static void step_solution  (void);
static void compare_solutions (void);
static void confirm_exit   (void);

/************************************************************************/
/* miscellaneous global variables                                       */
/************************************************************************/
 int in_the_exercise = 1;


/************************************************************************/
/* graphic initialization variables                                     */
/************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/************************************************************************/
/* This function is used for including drivers to the executable code   */
/************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/************************************************************************/
/* This fuction initializes the necessary graphical routines            */
/************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*********************************************************************/
  settext();
  /*********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
  else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }



/**********************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
   }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcclor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecoior);
          break;
   default : break;
   }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);    /* restore any hidden hot keys */
}


/**********************************************************************/
/* This function sets the text default values                        */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine  calls exer routine                                  */
/*********************************************************************/
void main()
{
  exer();
}
```

```
/**********************************************************************/
/* Routine that asks the question, then depending on the user's answer    */
/* makes necessary explanations                                            */
/**********************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  1");
  /*******************************************************************/
  outtextxy(10*x,2*y,"Construct an expression tree for the following  expression.");
  outtextxy(35*x,3*y,"a + b * c");
  /*******************************************************************/
 while (in_the_exercise == 1) {
  outtextxy(15*x,14*y,"Choose one of the following, as you need :");
  outtextxy(15*x,15*y,"    a) I'm done, I want to compare my solution with yours.");
  outtextxy(15*x,16*y,"    b) I want to see step by step solution.");
  outtextxy(15*x,17*y,"    c) This is enough for me, I want to exit.");
  outtextxy(15*x,18*y,"Enter your choice here --->");
  Ch = getch ();
  if(Ch==ESC) confirm_graph_exit();
   while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c'))) {
     outtextxy(48*x,18*y,"    Please type a, b, or c");
     Ch = getch ();
     if(Ch==ESC) confirm_graph_exit();
     if((Ch == 'a') || (Ch == 'b') || (Ch == 'c')) {
     setcolor(backcolor);
     bar(50*x,35*y/2,88*x,20*y);
     setcolor(forecolor);
     }
   }
     switch (Ch)          {
```

```
    case 'a': outtextxy(47*x,18*y,"a");
      outtextxy(52*x,18*y,"You want to compare your solu-");
      outtextxy(52*x,19*y,"tion with ours. So press any  ");
      outtextxy(52*x,20*y,"key to see it.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,35*y/2,179*x/2,22*y);
      bar(2*x,4*y,179*x/2,49*y/2);
      setcolor(forecolor);
      compare_solutions();
      break;
    case 'b': outtextxy(47*x,18*y,"b");
      outtextxy(52*x,18*y,"You want to see step by step");
      outtextxy(52*x,19*y,"solution. So press any key to ");
      outtextxy(52*x,20*y,"continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,35*y/2,179*x/2,22*y);
      bar(2*x,4*y,179*x/2,49*y/2);
      setcolor(forecolor);
      step_solution();
      break;
    case 'c': outtextxy(47*x,18*y,"c");
      confirm_exit();
      break;
    default  : break;
   }
  }
  closegraph();
}
```

```
/*************************************************************************/
/* This routine gives the solution to the exercise to be compared.      */
/*************************************************************************/
static void compare_solutions(void)
{

  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,4*y,179*x/2,49*y/2);
  /**********************************************************************/
  setcolor(forecolor);
  pieslice(40*x,10*y,0,359,2);   /* + */
  pieslice(35*x,12*y,0,359,2);   /* a */
  pieslice(45*x,12*y,0,359,2);   /* * */
  moveto(35*x,12*y); lineto(40*x,10*y); lineto(45*x,12*y);
  outtextxy(40*x,19*y/2,"+");
  outtextxy(35*x,25*y/2,"a");
  outtextxy(46*x,12*y,"*");
  pieslice(40*x,14*y,0,359,2);   /* b */
  pieslice(50*x,14*y,0,359,2);   /* c */
  moveto(40*x,14*y); lineto(45*x,12*y); lineto(50*x,14*y);
  outtextxy(39*x,29*y/2,"b");
  outtextxy(49*x,29*y/2,"c");
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field  again */
  bar(2*x,4*y,179*x/2,49*y/2);
  setcolor(forecolor);

}
```

```
/***********************************************************************/
/* This routine gives the step by step solution to the exercise        */
/***********************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(3*x/2,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
   /********************************************************************/
   pieslice(30*x,10*y,0,359,2);  /*    +    */
   pieslice(25*x,12*y,0,359,2);  /*    a    */
   pieslice(35*x,12*y,0,359,2);  /* (b * c) */
   moveto(25*x,12*y); lineto(30*x,10*y); lineto(35*x,12*y);
   outtextxy(30*x,19*y/2,"+");
   outtextxy(25*x,25*y/2,"a");
   outtextxy(32*x,25*y/2,"(b * c)");
   Pause(30*x,24*y);
   pieslice(50*x,10*y,0,359,2);  /* + */
   pieslice(45*x,12*y,0,359,2);  /* a */
   pieslice(55*x,12*y,0,359,2);  /* * */
   pieslice(50*x,14*y,0,359,2);  /* b */
   pieslice(60*x,14*y,0,359,2);  /* c */
   moveto(45*x,12*y); lineto(50*x,10*y); lineto(55*x,12*y);
   moveto(50*x,14*y); lineto(55*x,12*y); lineto(60*x,14*y);
   outtextxy(50*x,19*y/2,"+");
   outtextxy(45*x,25*y/2,"a");
   outtextxy(56*x,12*y,"*");
   outtextxy(49*x,29*y/2,"b");
   outtextxy(59*x,29*y/2,"c");
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field  again */
   bar(2*x,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
}
```

```
/*************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,22*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,21*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
    switch (ch)         {
    case 'y': in_the_exercise = 0;
            break;
    case 'Y': in_the_exercise = 0;
            break;

    case 'n': setcolor(backcolor);
            bar(46*x,35*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;

    case 'N': setcolor(backcolor);
            bar(46*x,35*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;

    default : break;
    }
}
```

```
/* PROGRAM    : q452.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 2, 1990
   REVISED    : Apr. 2, 1990

   DESCRIPTION : This program contains the second exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"


#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk           find_t
  #define ff_name         name
#elif defined(__ZTC__)                   /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
static void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution   (void);
static void compare_solutions (void);
static void confirm_exit    (void);

/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;


/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code     */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines              */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*******************************************************************/
  settext();
  /*******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
      ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
        setfillstyle(SOLID_FILL,BLACK);
        backcolor = BLACK;
        quitcolor = WHITE;
      }
    else {
        setfillstyle(SOLID_FILL,BLUE);
        backcolor = BLUE;
        quitcolor = RED;
      }
   forecolor = WHITE;
  }


/***********************************************************************/
/* This function sets the text default values                          */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen               */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }
```

```
/************************************************************************/
/* Routine that asks the question, then depending on the user's answer     */
/* makes necessary explanations                                              */
/************************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  2");
  /**********************************************************************/
  outtextxy(10*x,2*y,"Construct an expression tree for the following  expression.");
  outtextxy(30*x,3*y,"((a - b) / c) * (d + e / f)");
  /**********************************************************************/
  while (in_the_exercise == 1) {
  outtextxy(15*x,14*y,"Choose one of the following, as you need :");
  outtextxy(15*x,15*y,"    a) I'm done, I want to compare my solution with yours.");
  outtextxy(15*x,16*y,"    b) I want to see step by step solution.");
  outtextxy(15*x,17*y,"    c) This is enough for me, I want to exit.");
  outtextxy(15*x,18*y,"Enter your choice here --->");
  Ch = getch ();
  if(Ch==ESC) confirm_graph_exit();
   while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,18*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,35*y/2,88*x,20*y);
    setcolor(forecolor);
    }
   }
    switch (Ch)         {
```

```
    case 'a': outtextxy(47*x,18*y,"a");
      outtextxy(52*x,18*y,"You want to compare your solu-");
      outtextxy(52*x,19*y,"tion with ours. So press any  ");
      outtextxy(52*x,20*y,"key to see it.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,35*y/2,179*x/2,22*y);
      bar(2*x,4*y,179*x/2,49*y/2);
      setcolor(forecolor);
      compare_solutions();
      break;
    case 'b': outtextxy(47*x,18*y,"b");
      outtextxy(52*x,18*y,"You want to see step by step");
      outtextxy(52*x,19*y,"solution. So press any key to ");
      outtextxy(52*x,20*y,"continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,35*y/2,179*x/2,22*y);
      bar(2*x,4*y,179*x/2,49*y/2);
      setcolor(forecolor);
      step_solution();
      break;
    case 'c': outtextxy(47*x,18*y,"c");
      confirm_exit();
      break;
    default  : break;
   }
 }
  closegraph();
}
```

```
/**********************************************************************/
/* This routine gives the solution to the exercise to be compared.       */
/**********************************************************************/
static void compare_solutions(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,4*y,179*x/2,49*y/2);
  /*******************************************************************/
  setcolor(forecolor);
  pieslice(40*x,8*y,0,359,2);   /* * */
  pieslice(35*x,10*y,0,359,2);  /* / */
  pieslice(45*x,10*y,0,359,2);  /* + */
  moveto(35*x,10*y); lineto(40*x,8*y); lineto(45*x,10*y);
  outtextxy(40*x,15*y/2,"*");
  outtextxy(33*x,10*y,"/");
  outtextxy(46*x,10*y,"+");
  /*******************************************************************/
  pieslice(42*x,12*y,0,359,2);   /* d */
  pieslice(50*x,12*y,0,359,2);   /* / */
  moveto(42*x,12*y); lineto(45*x,10*y); lineto(50*x,12*y);
  pieslice(30*x,12*y,0,359,2);   /* - */
  pieslice(38*x,12*y,0,359,2);   /* c */
  moveto(30*x,12*y); lineto(35*x,10*y); lineto(38*x,12*y);
  outtextxy(28*x,12*y,"-");
  outtextxy(38*x,25*y/2,"c");
  outtextxy(42*x,25*y/2,"d");
  outtextxy(51*x,12*y,"/");
  /*******************************************************************/
  pieslice(45*x,14*y,0,359,2);   /* e */
  pieslice(55*x,14*y,0,359,2);   /* f */
  moveto(45*x,14*y); lineto(50*x,12*y); lineto(55*x,14*y);
  pieslice(25*x,14*y,0,359,2);   /* a */
  pieslice(35*x,14*y,0,359,2);   /* b */
  moveto(25*x,14*y); lineto(30*x,12*y); lineto(35*x,14*y);
  outtextxy(25*x,29*y/2,"a");
  outtextxy(35*x,29*y/2,"b");
```

```
   outtextxy(45*x,29*y/2,"e");
   outtextxy(55*x,29*y/2,"f");
   /*****************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);         /* Clean the game field  again */
   bar(2*x,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
}



/*******************************************************************/
/* This routine gives the step by step solution to the exercise               */
/*******************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);         /* Clean the game field */
   bar(3*x/2,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   setcolor(forecolor);
   pieslice(20*x,6*y,0,359,2);   /*      *      */
   pieslice(15*x,8*y,0,359,2);  /* ((a - b) / c  */
   pieslice(25*x,8*y,0,359,2);  /* (d + e / f)   */
   moveto(15*x,8*y);  lineto(20*x,6*y);  lineto(25*x,8*y);
   outtextxy(20*x,11*y/2,"*");
   outtextxy(5*x,17*y/2,"((a - b) / c)");
   outtextxy(21*x,17*y/2,"(d + e / f)");
   Pause(30*x,24*y);
   /*****************************************************************/
   pieslice(60*x,6*y,0,359,2);   /* * */
   pieslice(55*x,8*y,0,359,2);   /* / */
   pieslice(65*x,8*y,0,359,2);   /* + */
   moveto(55*x,8*y);  lineto(60*x,6*y);  lineto(65*x,8*y);
   outtextxy(60*x,11*y/2,"*");
   outtextxy(53*x,8*y,"/");
```

```
/****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
   switch (ch)         {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);      /* restore any hidden hot keys */
}


/*********************************************************************/
/* main routine that calls exer routine                              */
/*********************************************************************/
void main()
{
  exer();
}
```

```
outtextxy(66*x,8*y,"+");
/*****************************************************************/
pieslice(62*x,10*y,0,359,2);  /*  d  */
pieslice(70*x,10*y,0,359,2);  /* (e / f) */
moveto(62*x,10*y); lineto(65*x,8*y); lineto(70*x,10*y);
pieslice(50*x,10*y,0,359,2);  /* (a - b) */
pieslice(58*x,10*y,0,359,2);  /*  c  */
moveto(50*x,10*y); lineto(55*x,8*y); lineto(58*x,10*y);
outtextxy(46*x,21*y/2,"(a - b)");
outtextxy(58*x,21*y/2,"c");
outtextxy(62*x,21*y/2,"d");
outtextxy(67*x,21*y/2,"(e / f)");
Pause(30*x,24*y);
/*****************************************************************/
pieslice(40*x,12*y,0,359,2);  /* * */
pieslice(35*x,14*y,0,359,2);  /* / */
pieslice(45*x,14*y,0,359,2);  /* + */
moveto(35*x,14*y); lineto(40*x,12*y); lineto(45*x,14*y);
outtextxy(40*x,23*y/2,"*");
outtextxy(33*x,14*y,"/");
outtextxy(46*x,14*y,"+");
/*****************************************************************/
pieslice(42*x,16*y,0,359,2);  /* d */
pieslice(50*x,16*y,0,359,2);  /* / */
moveto(42*x,16*y); lineto(45*x,14*y); lineto(50*x,16*y);
pieslice(30*x,16*y,0,359,2);  /* - */
pieslice(38*x,16*y,0,359,2);  /* c */
moveto(30*x,16*y); lineto(35*x,14*y); lineto(38*x,16*y);
outtextxy(28*x,16*y,"-");
outtextxy(38*x,33*y/2,"c");
outtextxy(42*x,33*y/2,"d");
outtextxy(51*x,16*y,"/");
/*****************************************************************/
pieslice(45*x,18*y,0,359,2);  /* e */
pieslice(55*x,18*y,0,359,2);  /* f */
moveto(45*x,18*y); lineto(50*x,16*y); lineto(55*x,18*y);
```

```
   pieslice(25*x,18*y,0,359,2);   /* a */
   pieslice(35*x,18*y,0,359,2);   /* b */
   moveto(25*x,18*y); lineto(30*x,16*y); lineto(35*x,18*y);
   outtextxy(25*x,37*y/2,"a");
   outtextxy(35*x,37*y/2,"b");
   outtextxy(45*x,37*y/2,"e");
   outtextxy(55*x,37*y/2,"f");
   /*****************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field  again */
   bar(2*x,4*y,179*x/2,49*y/2);
   setcolor(forecolor);

}

/*********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,22*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,21*y,179*x/2,49*y/2);
       setcolor(forecolor);
    }
    switch (ch)        {
     case 'y': in_the_exercise = 0;
           break;
     case 'Y': in_the_exercise = 0;
```

```
                break;

          case 'n': setcolor(backcolor);
                bar(46*x,35*y/2,179*x/2,22*y);
                setcolor(forecolor);
                break;

          case 'N': setcolor(backcolor);
                bar(46*x,35*y/2,179*x/2,22*y);
                setcolor(forecolor);
                break;

          default : break;
          }
     }
```

```
/* PROGRAM   : q453.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 4, 1990
   REVISED   : Apr. 4, 1990

   DESCRIPTION : This program contains the third exercise about the binary
                 trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name          name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer          (void);

/**********************************************************************/
/* graphic initialization variables                                  */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/**********************************************************************/
/* This function is used for including drivers to the executable code        */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/***********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/***********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
       }
  forecolor = WHITE;
  }
```

```
/***********************************************************************/
/* This function sets the text default values                         */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}

/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }

/***********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
```

```
    ch = getch ();
    if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
    setcolor(backcolor);
    bar(31*x,23*y,69*x,97*y/4);
    setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;

   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;

   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;

   case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;

   default : break;
   }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* main routine that calls exer routine                                   */
/*************************************************************************/
void main()
{
  exer();
}

/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                            */
/*************************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  3");
   outtextxy(20*x,2*y,"Consider the following binary tree.");
   /*********************************************************************/
   pieslice(40*x,3*y,0,359,2);
   pieslice(35*x,5*y,0,359,2);
   pieslice(45*x,5*y,0,359,2);
   pieslice(30*x,7*y,0,359,2);
   pieslice(38*x,7*y,0,359,2);
   pieslice(50*x,7*y,0,359,2);
   pieslice(25*x,9*y,0,359,2);
   pieslice(35*x,9*y,0,359,2);
   pieslice(42*x,9*y,0,359,2);
   pieslice(46*x,9*y,0,359,2);
   pieslice(55*x,9*y,0,359,2);
   pieslice(20*x,11*y,0,359,2);
   pieslice(30*x,11*y,0,359,2);
pieslice(38*x,11*y,0,359,2);
pieslice(42*x,11*y,0,359,2);
pieslice(50*x,11*y,0,359,2);
pieslice(60*x,11*y,0,359,2);
moveto(20*x,11*y); lineto(25*x,9*y);
lineto(30*x,7*y);  lineto(35*x,5*y);
lineto(40*x,3*y);  lineto(45*x,5*y);
lineto(50*x,7*y);  lineto(55*x,9*y);
lineto(60*x,11*y);
moveto(30*x,11*y); lineto(35*x,9*y);
lineto(38*x,7*y);  lineto(42*x,9*y);
lineto(38*x,11*y);
moveto(35*x,5*y); lineto(38*x,7*y);
moveto(42*x,11*y); lineto(46*x,9*y);
lineto(50*x,11*y);
moveto(46*x,9*y); lineto(50*x,7*y);
outtextxy(79*x/2,5*y/2,"A");
outtextxy(33*x,5*y,"B");
outtextxy(46*x,5*y,"C");
outtextxy(28*x,7*y,"D");
outtextxy(39*x,7*y,"E");
outtextxy(51*x,7*y,"F");
outtextxy(23*x,9*y,"G");
outtextxy(33*x,9*y,"H");
outtextxy(43*x,9*y,"I");
outtextxy(47*x,9*y,"J");
outtextxy(56*x,9*y,"K");
outtextxy(20*x,23*y/2,"L");
outtextxy(30*x,23*y/2,"M");
outtextxy(38*x,23*y/2,"N");
outtextxy(42*x,23*y/2,"O");
outtextxy(50*x,23*y/2,"P");
outtextxy(60*x,23*y/2,"Q");
/****************************************************************************/
outtextxy(18*x,13*y,"Which one of the following statements is true ?");
outtextxy(20*x,15*y,"a) H and M forms the left subtree of vertex B");
outtextxy(20*x,16*y,"b)  J is the root of left subtree ofvertex F ");
outtextxy(20*x,17*y,"c)  N is the right subtree of vertex E");
outtextxy(20*x,18*y,"d)  All of the above statements are correct");
outtextxy(18*x,20*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,20*y,"    Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(50*x,19*y,88*x,19*y);
   setcolor(forecolor);
 }
 switch (Ch)          {
 case 'a': outtextxy(50*x,20*y,"a");
      outtextxy(55*x,20*y,"Sorry, that's not true!");
      outtextxy(55*x,21*y,"because, they form the");
      outtextxy(55*x,22*y,"left subtree of vertex E.");
      outtextxy(55*x,23*y,"The answer is 'b'.");
      break;

 case 'b': outtextxy(50*x,20*y,"b");
      outtextxy(55*x,20*y,"Correct. You are doing fine!");
      break;

 case 'c': outtextxy(50*x,20*y,"c");
      outtextxy(55*x,20*y,"No. N is the left subtree of");
      outtextxy(55*x,21*y,"the vertex I. The answer is");
      outtextxy(55*x,22*y,"'b'.");
      break;

 case 'd': outtextxy(50*x,20*y,"d");
      outtextxy(55*x,20*y,"No. Because, if you carefully");
      outtextxy(55*x,21*y,"examine, you'll see that 'a'");
```

```
        outtextxy(55*x,22*y,"and 'c' is wrong, so 'd' is.");
        outtextxy(55*x,23*y,"The answer is 'b'");
        break;

    default : break;
    }
    Pause(15*x,24*y);
    closegraph();
}
```

```
/* PROGRAM   : q454.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 4, 1990
   REVISED   : Apr. 4, 1990

   DESCRIPTION : This program contains the fourth exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                      /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)    /* MSC/QuickC */
  #define bioskey(a)       _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)      _dos_findnext(a)
  #define ffblk            find_t
  #define ff_name          name
#elif defined( __ZTC__)                    /* Zortech C/C++ */
  #define ffblk            FIND
  #define ff_name          name
  #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);
static void error_exit    (int errnum);

/* tutorial functions     */
static void exer              (void);
static void example           (void);
static void show_alg          (void);
static void step_solution     (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/***********************************************************************/
/* miscellaneous global variables                                      */
/***********************************************************************/
 int in_the_exercise = 1;

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code          */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines                  */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /******************************************************************/
  settext();
  /******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }


/*********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)          {
```

```
    case 'y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'Y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'n': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    case 'N': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}


/***********************************************************************/
/* This function sets the text default values                          */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen           */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
 }


/*********************************************************************/
/* main routine   calls exer routine                                 */
/*********************************************************************/
void main()
{
  exer();
}
```

```
/***********************************************************************/
/* Routine that asks the question, then depending on the user's answer      */
/* makes necessary explanations                                              */
/***********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  4");
   /*******************************************************************/
   outtextxy(2*x,2*y,"Give the preorder listing of the vertices for the following binary
                        tree.");
   /*******************************************************************/
   pieslice(40*x,3*y,0,359,2);
   pieslice(35*x,5*y,0,359,2);
   pieslice(45*x,5*y,0,359,2);
   pieslice(30*x,7*y,0,359,2);
   pieslice(38*x,7*y,0,359,2);
   pieslice(50*x,7*y,0,359,2);
   pieslice(25*x,9*y,0,359,2);
   pieslice(35*x,9*y,0,359,2);
   pieslice(42*x,9*y,0,359,2);
   pieslice(46*x,9*y,0,359,2);
   pieslice(55*x,9*y,0,359,2);
   pieslice(20*x,11*y,0,359,2);
   pieslice(30*x,11*y,0,359,2);
   pieslice(38*x,11*y,0,359,2);
   pieslice(42*x,11*y,0,359,2);
   pieslice(50*x,11*y,0,359,2);
   pieslice(60*x,11*y,0,359,2);
   moveto(20*x,11*y);  lineto(25*x,9*y);
   lineto(30*x,7*y);  lineto(35*x,5*y);
lineto(40*x,3*y);  lineto(45*x,5*y);
lineto(50*x,7*y);  lineto(55*x,9*y);
lineto(60*x,11*y);
moveto(30*x,11*y); lineto(35*x,9*y);
lineto(38*x,7*y);  lineto(42*x,9*y);
lineto(38*x,11*y);
moveto(35*x,5*y); lineto(38*x,7*y);
moveto(42*x,11*y); lineto(46*x,9*y);
lineto(50*x,11*y);
moveto(46*x,9*y); lineto(50*x,7*y);
outtextxy(79*x/2,5*y/2,"A");
outtextxy(33*x,5*y,"B");
outtextxy(46*x,5*y,"C");
outtextxy(28*x,7*y,"D");
outtextxy(39*x,7*y,"E");
outtextxy(51*x,7*y,"F");
outtextxy(23*x,9*y,"G");
outtextxy(33*x,9*y,"H");
outtextxy(43*x,9*y,"I");
outtextxy(47*x,9*y,"J");
outtextxy(56*x,9*y,"K");
outtextxy(20*x,23*y/2,"L");
outtextxy(30*x,23*y/2,"M");
outtextxy(38*x,23*y/2,"N");
outtextxy(42*x,23*y/2,"O");
outtextxy(50*x,23*y/2,"P");
outtextxy(60*x,23*y/2,"Q");
/**********************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"     a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"     b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"     c) I want to see step by step solution.");
outtextxy(15*x,18*y,"     d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
```

```
if(Ch==ESC) confirm_graph_exit();
 while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,19*y,"   Please type a, b, c or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
   setcolor(backcolor);
   bar(50*x,37*y/2,88*x,20*y);
   setcolor(forecolor);
   }
 }
   switch (Ch)          {
   case 'a': outtextxy(47*x,19*y,"a");
     outtextxy(52*x,19*y,"You want to see the algorithm ");
     outtextxy(52*x,20*y,"again. Press any key to continue.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,37*y/2,179*x/2,21*y);
     bar(2*x,13*y,179*x/2,49*y/2);
     setcolor(forecolor);
     show_alg();
     break;
  case 'b': outtextxy(47*x,19*y,"b");
     outtextxy(52*x,19*y,"You want to compare your solu-");
     outtextxy(52*x,20*y,"tion with ours. So press any  ");
     outtextxy(52*x,21*y,"key to see it.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,37*y/2,179*x/2,22*y);
     bar(2*x,13*y,179*x/2,49*y/2);
     setcolor(forecolor);
     compare_solutions();
     break;
  case 'c': outtextxy(47*x,19*y,"c");
     outtextxy(52*x,19*y,"You want to see step by step");
     outtextxy(52*x,20*y,"solution. So press any key to ");
```

```
      outtextxy(52*x,21*y,"continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,22*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      step_solution();
      break;
    case 'd': outtextxy(47*x,19*y,"d");
      confirm_exit();
      break;
    default  : break;
    }
  }
  closegraph();
}

/************************************************************************/
/* This routine gives preorder traversal of a binary tree algorithm                */
/************************************************************************/
static void show_alg(void)
{
    outtextxy(15*x,14*y,"PREORDER TRAVERSAL ALGORITHM OF A BINARY
TREE");
  outtextxy(2*x,15*y,"Step 1 (visit)    Visit the root.");
   outtextxy(2*x,16*y,"Step 2 (go left)  Go to the left subtree, if one  exists, do a pre-
order");
  outtextxy(2*x,17*y,"       traversal.");
   outtextxy(2*x,18*y,"Step 3 (go right) Go to the right subtree, if one exists, and do
a preorder");
  outtextxy(2*x,19*y,"       traversal.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/*********************************************************************/
/* This routine gives the solution to the exercise to be compared.   */
/*********************************************************************/
static void compare_solutions(void)
{
   setcolor(backcolor);        /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   /******************************************************************/
   outtextxy(3*x,29*y/2,"Preorder  Listing");
   moveto(2*x,15*y);  lineto(60*x,15*y);
   outtextxy(3*x,16*y,"A, B, D, G, L, E, H, M, I, N, C, F, J, O, P, K, Q");
   /******************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);        /* Clean the game field again */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}

/*********************************************************************/
/* This routine gives the step by step solution to the exercise      */
/*********************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);        /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   outtextxy(3*x,29*y/2,"Preorder  Listing");
   moveto(2*x,15*y);  lineto(57*x,15*y);
   /******************************************************************/
   outtextxy(3*x,16*y,"A");    /* Visit the root A */
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,50*x,49*y/2);
   setcolor(forecolor);
```

```
/******************************************************************/
/* Go to left subtree, and do a preorder traversal again.        */
/******************************************************************/
setcolor(backcolor);
moveto(40*x,3*y); lineto(35*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,3*y); lineto(35*x,5*y);    /* Visit B */
setlinestyle(0,0,3);
outtextxy(6*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/******************************************************************/
/* Go to left subtree, and do a preorder traversal again.        */
/******************************************************************/
setcolor(backcolor);
moveto(35*x,5*y); lineto(30*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,7*y); lineto(35*x,5*y);    /* Visit D */
setlinestyle(0,0,3);
outtextxy(9*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/******************************************************************/
/* Go to left subtree, and do a preorder traversal again.        */
/******************************************************************/
setcolor(backcolor);
moveto(25*x,9*y); lineto(30*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,7*y); lineto(25*x,9*y);    /* Visit G */
```

```
setlinestyle(0,0,3);
outtextxy(12*x,16*y,"G");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
/* Go to left subtree, and do a preorder traversal again.            */
/********************************************************************/
setcolor(backcolor);
moveto(25*x,9*y); lineto(20*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,11*y); lineto(25*x,9*y);    /* Visit L */
setlinestyle(0,0,3);
outtextxy(15*x,16*y,"L");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
/* Since L is a terminal node, it does not have a subtree, so go back */
/* until the vertex B, and go to its right subtree, and visit root E  */
/********************************************************************/
setcolor(backcolor);
moveto(35*x,5*y); lineto(38*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,5*y); lineto(38*x,7*y);    /* Visit E */
setlinestyle(0,0,3);
outtextxy(18*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
```

```
/**********************************************************************/
/* Go to left subtree of the vertex E.                              */
/**********************************************************************/
setcolor(backcolor);
moveto(35*x,9*y); lineto(38*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,9*y); lineto(38*x,7*y);    /* Visit H */
setlinestyle(0,0,3);
outtextxy(21*x,16*y,"H");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
/* Go to left subtree of the vertex H.                              */
/**********************************************************************/
setcolor(backcolor);
moveto(30*x,11*y); lineto(35*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,11*y); lineto(35*x,9*y);    /* Visit M */
setlinestyle(0,0,3);
outtextxy(24*x,16*y,"M");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/**********************************************************************/
/* Since M is a terminal node, it does not have a subtree, so go back */
/* until the vertex E, and go to its right subtree, and visit root I  */
/**********************************************************************/
setcolor(backcolor);
moveto(42*x,9*y); lineto(38*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
```

```
moveto(42*x,9*y);  lineto(38*x,7*y);    /* Visit I */
setlinestyle(0,0,3);
outtextxy(27*x,16*y,"I");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
/* Go to left subtree of the vertex I.                             */
/*******************************************************************/
setcolor(backcolor);
moveto(38*x,11*y);  lineto(42*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,11*y);  lineto(42*x,9*y);    /* Visit N */
setlinestyle(0,0,3);
outtextxy(30*x,16*y,"N");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
/* Since N is a terminal node, it does not have a subtree, so go back  */
/* until the vertex A, and go to its right subtree, and visit root C.  */
/*******************************************************************/
setcolor(backcolor);
moveto(40*x,3*y);  lineto(45*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,3*y);  lineto(45*x,5*y);    /* Visit C */
setlinestyle(0,0,3);
outtextxy(33*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
```

```
/*********************************************************************/
/* The vertex C does not have a left subtree so, go to its right subtree        */
/*********************************************************************/
setcolor(backcolor);
moveto(50*x,7*y);  lineto(45*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,7*y);  lineto(45*x,5*y);    /* Visit F */
setlinestyle(0,0,3);
outtextxy(36*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
/* Go to left subtree of the vertex F.                                          */
/*********************************************************************/
setcolor(backcolor);
moveto(50*x,7*y);  lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,7*y);  lineto(46*x,9*y);    /* Visit J */
setlinestyle(0,0,3);
outtextxy(39*x,16*y,"J");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
/* Go to left subtree of the vertex J.                                          */
/*********************************************************************/
setcolor(backcolor);
moveto(42*x,11*y);  lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(42*x,11*y);  lineto(46*x,9*y);    /* Visit O */
```

```
setlinestyle(0,0,3);
outtextxy(42*x,16*y,"O");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* Since O is a terminal node, it does not have a subtree, so go back          */
/* to the vertex J, and go to its right subtree, and visit the vertex P.       */
/*****************************************************************/
setcolor(backcolor);
moveto(50*x,11*y); lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,11*y); lineto(46*x,9*y);    /* Visit P */
setlinestyle(0,0,3);
outtextxy(45*x,16*y,"P");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* Since P is a terminal node, it does not have a subtree, so go back          */
/* to the vertex F, and go to its right subtree, and visit the root K          */
/*****************************************************************/
setcolor(backcolor);
moveto(50*x,7*y); lineto(55*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,7*y); lineto(55*x,9*y);    /* Visit K */
setlinestyle(0,0,3);
outtextxy(48*x,16*y,"K");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
```

```
/*********************************************************************/
/* The vertex K does not have a left subtree so, go to its right subtree        */
/*********************************************************************/
setcolor(backcolor);
moveto(55*x,9*y); lineto(60*x,11*y),
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,9*y); lineto(60*x,11*y);    /* Visit Q */
setlinestyle(0,0,3);
outtextxy(51*x,16*y."Q");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
/*********************************************************************/
/* Clean the game field  again */
bar(3*x/2,47*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/*********************************************************************/
/*              Redraw the tree                                                  */
/*********************************************************************/
moveto(20*x,11*y); lineto(25*x,9*y);
lineto(30*x,7*y);  lineto(35*x,5*y);
lineto(40*x,3*y);  lineto(45*x,5*y);
lineto(50*x,7*y);  lineto(55*x,9*y);
lineto(60*x,11*y);
moveto(30*x,11*y); lineto(35*x,9*y);
lineto(38*x,7*y);  lineto(42*x,9*y);
lineto(38*x,11*y);
moveto(35*x,5*y);  lineto(38*x,7*y);
moveto(42*x,11*y); lineto(46*x,9*y);
lineto(50*x,11*y);
moveto(46*x,9*y); lineto(50*x,7*y);
}
```

```
/**********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
    }
    switch (ch)          {
    case 'y': in_the_exercise = 0;
          break;
    case 'Y': in_the_exercise = 0;
          break;

    case 'n': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    case 'N': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    default : break;
    }
}
```

```
/* PROGRAM   : q455.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 4, 1990
   REVISED    : Apr. 4, 1990

   DESCRIPTION : This program contains the fifth exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);
static void error_exit    (int errnum);

/* tutorial functions      */
static void exer              (void);
static void example           (void);
static void show_alg          (void);
static void step_solution     (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
 int in_the_exercise = 1;

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code        */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines              */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /******************************************************************/
  settext();
  /******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
    else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
   forecolor = WHITE;
 }

/***************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL); /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;

     case 'Y': closegraph();
            videoinit();
            exit(0);
            break;

     case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;

     case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;

     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}
/*********************************************************************/
/* This function sets the text default values                        */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/************************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/************************************************************************/
/* main routine that calls exer routine                                */
/************************************************************************/
void main()
{
  exer();
}
```

```
/***********************************************************************/
/* Routine that asks the question, then depending on the user's answer      */
/* makes necessary explanations                                             */
/***********************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  5");
  /*********************************************************************/
  outtextxy(2*x,2*y,"Give the preorder listing of the vertices for the following binary
                    tree.");
  /*********************************************************************/
  pieslice(40*x,3*y,0,359,2);
  pieslice(30*x,5*y,0,359,2);
  pieslice(50*x,5*y,0,359,2);
  pieslice(25*x,7*y,0,359,2);
  pieslice(35*x,7*y,0,359,2);
  pieslice(45*x,7*y,0,359,2);
  pieslice(55*x,7*y,0,359,2);
  pieslice(20*x,9*y,0,359,2);
  pieslice(30*x,9*y,0,359,2);
  pieslice(40*x,9*y,0,359,2);
  pieslice(50*x,9*y,0,359,2);
  pieslice(60*x,9*y,0,359,2);
  moveto(20*x,9*y); lineto(25*x,7*y);
  lineto(30*x,5*y); lineto(40*x,3*y);
  lineto(50*x,5*y); lineto(55*x,7*y);
  lineto(60*x,9*y);
  moveto(30*x,9*y); lineto(35*x,7*y);
  lineto(40*x,9*y);
  moveto(30*x,5*y); lineto(35*x,7*y);
moveto(50*x,5*y); lineto(45*x,7*y);
lineto(50*x,9*y);
outtextxy(79*x/2,5*y/2,"A");
outtextxy(28*x,5*y,"B");
outtextxy(51*x,5*y,"C");
outtextxy(23*x,7*y,"D");
outtextxy(36*x,7*y,"E");
outtextxy(43*x,7*y,"F");
outtextxy(56*x,7*y,"G");
outtextxy(20*x,19*y/2,"H");
outtextxy(30*x,19*y/2,"I");
outtextxy(40*x,19*y/2,"J");
outtextxy(50*x,19*y/2,"K");
outtextxy(60*x,19*y/2,"L");
/*****************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
/*****************************************************************/
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"   Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
```

```
switch (Ch)          {
case 'a': outtextxy(47*x,19*y,"a");
  outtextxy(52*x,19*y,"You want to see the algorithm ");
  outtextxy(52*x,20*y,"again. Press any key to continue.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(50*x,37*y/2,179*x/2,21*y);
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  show_alg();
  break;
case 'b': outtextxy(47*x,19*y,"b");
  outtextxy(52*x,19*y,"You want to compare your solu-");
  outtextxy(52*x,20*y,"tion with ours. So press any  ");
  outtextxy(52*x,21*y,"key to see it.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(50*x,37*y/2,179*x/2,22*y);
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  compare_solutions();
  break;
case 'c': outtextxy(47*x,19*y,"c");
  outtextxy(52*x,19*y,"You want to see step by step");
  outtextxy(52*x,20*y,"solution. So press any key to ");
  outtextxy(52*x,21*y,"continue.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(50*x,37*y/2,179*x/2,22*y);
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  step_solution();
  break;
case 'd': outtextxy(47*x,19*y,"d");
  confirm_exit();
  break;
```

```
      default  : break;
     }
  }
   closegraph();
}
```

```
/**********************************************************************/
/* This routine gives preorder traversal of a binary tree algorithm                    */
/**********************************************************************/
static void show_alg(void)
{
     outtextxy(15*x,14*y,"PREORDER TRAVERSAL ALGORITHM OF A BINARY
TREE");
   outtextxy(2*x,15*y,"Step 1 (visit)    Visit the root.");
   outtextxy(2*x,16*y,"Step 2 (go left)  Go to the left subtree, if one  exists, do a pre-
                                    order");
   outtextxy(2*x,17*y,"      traversal.");
   outtextxy(2*x,18*y,"Step 3 (go right) Go to the right subtree, if one exists, and do
                                    a preorder");
   outtextxy(2*x,19*y,"      traversal.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}
```

```
/***********************************************************************/
/* This routine gives the solution to the exercise to be compared.      */
/***********************************************************************/
static void compare_solutions(void)
{

   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   /*******************************************************************/
   outtextxy(3*x,29*y/2,"Preorder  Listing");
   moveto(2*x,15*y);  lineto(42*x,15*y);
   outtextxy(3*x,16*y,"A, B, D, H, E, I, J, C, F, K, G, L");
   /*******************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field again */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}


/***********************************************************************/
/* This routine gives the step by step solution to the exercise        */
/***********************************************************************/
static void step_solution(void)
{

   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   outtextxy(3*x,29*y/2,"Preorder  Listing");
   moveto(2*x,15*y);  lineto(42*x,15*y);
   /*******************************************************************/
   outtextxy(3*x,16*y,"A");      /* Visit the root A */
   Pause(30*x,24*y);
   setcolor(backcolor);
```

```
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
/* Go to left subtree, and do a preorder traversal again.          */
/*******************************************************************/
setcolor(backcolor);
moveto(40*x,3*y); lineto(30*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,3*y); lineto(30*x,5*y);    /* Visit B */
setlinestyle(0,0,3);
outtextxy(6*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
/* Go to left subtree, and do a preorder traversal again.          */
/*******************************************************************/
setcolor(backcolor);
moveto(30*x,5*y); lineto(25*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,5*y); lineto(25*x,7*y);    /* Visit D */
setlinestyle(0,0,3);
outtextxy(9*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
/* Go to left subtree, and do a preorder traversal again.          */
/*******************************************************************/
setcolor(backcolor);
moveto(25*x,7*y); lineto(20*x,9*y);
setlinestyle(3,0,3);
```

```
setcolor(forecolor);
moveto(25*x,7*y); lineto(20*x,9*y);    /* Visit H */
setlinestyle(0,0,3);
outtextxy(12*x,16*y,"H");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* Since H is a terminal node, it does not have a subtree, so go back          */
/* until the vertex B, and go to its right subtree, and visit root E            */
/*****************************************************************/
setcolor(backcolor);
moveto(30*x,5*y); lineto(35*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,5*y); lineto(35*x,7*y);    /* Visit E */
setlinestyle(0,0,3);
outtextxy(15*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* Go to left subtree of the vertex E.                                          */
/*****************************************************************/
setcolor(backcolor);
moveto(35*x,7*y); lineto(30*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y); lineto(30*x,9*y);    /* Visit I */
setlinestyle(0,0,3);
outtextxy(18*x,16*y,"I");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
```

```
setcolor(forecolor);
/*****************************************************************/
/* Since I is a terminal node, it does not have a subtree, so go back          */
/* until the vertex E, and go to its right subtree, and visit root J            */
/*****************************************************************/
setcolor(backcolor);
moveto(35*x,7*y); lineto(40*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y); lineto(40*x,9*y);    /* Visit J */
setlinestyle(0,0,3);
outtextxy(21*x,16*y,"J");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* We have completed traversal of the left subtree of the root A. We now       */
/* will visit the right subtree of A.                                           */
/*****************************************************************/
setcolor(backcolor);
moveto(40*x,3*y); lineto(50*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,3*y); lineto(50*x,5*y);    /* Visit C */
setlinestyle(0,0,3);
outtextxy(24*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* Go to left subtree of the vertex C.                                          */
/*****************************************************************/
setcolor(backcolor);
moveto(50*x,5*y); lineto(45*x,7*y);
```

```
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,5*y);  lineto(45*x,7*y);    /* Visit F */
setlinestyle(0,0,3);
outtextxy(27*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
/* The vertex F does not have a left subtree so, go to its right subtree       */
/*********************************************************************/
setcolor(backcolor);
moveto(50*x,9*y);  lineto(45*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,9*y);  lineto(45*x,7*y);    /* Visit K */
setlinestyle(0,0,3);
outtextxy(30*x,16*y,"K");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
/* Since K is a terminal node, it does not have a subtree, so go back          */
/* until the vertex C, and go to its right subtree, and visit root G           */
/*********************************************************************/
setcolor(backcolor);
moveto(50*x,5*y);  lineto(55*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,5*y);  lineto(55*x,7*y);    /* Visit G */
setlinestyle(0,0,3);
outtextxy(33*x,16*y,"G");
Pause(30*x,24*y);
setcolor(backcolor);
```

```
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /*********************************************************************/
    /* The vertex G does not have a left subtree so, go to its right subtree       */
    /*********************************************************************/
    setcolor(backcolor);
    moveto(55*x,7*y);  lineto(60*x,9*y);
    setlinestyle(3,0,3);
    setcolor(forecolor);
    moveto(55*x,7*y);  lineto(60*x,9*y);     /* Visit L */
    setlinestyle(0,0,3);
    outtextxy(36*x,16*y,"L");
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /*********************************************************************/
    /* Clean the game field  again */
    bar(3*x/2,47*y/4,179*x/2,49*y/2);
    setcolor(forecolor);
    /*********************************************************************/
    /*                Redraw the tree                                          */
    /*********************************************************************/
    moveto(20*x,9*y);  lineto(25*x,7*y);
    lineto(30*x,5*y);  lineto(40*x,3*y);
    lineto(50*x,5*y);  lineto(55*x,7*y);
    lineto(60*x,9*y);
    moveto(30*x,9*y);  lineto(35*x,7*y);
    lineto(40*x,9*y);
    moveto(30*x,5*y);  lineto(35*x,7*y);
    moveto(50*x 5*y);  lineto(45*x,7*y);
    lineto(50*x,9*y),
}
```

```
/*********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,23*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,22*y,179*x/2,49*y/2);
       setcolor(forecolor);
    }
    switch (ch)          {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;

    case 'n': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    case 'N': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    default : break;
     }
}
```

```
/* PROGRAM   : q456.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 4, 1990
   REVISED    : Apr. 4, 1990

   DESCRIPTION : This program contains the sixth exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)     _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)    _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name        name
#elif defined(__ZTC__)                   /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name        name
   #define ff_attrib      attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);
static void example          (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/**********************************************************************/
/* miscellaneous global variables                                      */
/**********************************************************************/
 int in_the_exercise = 1;


/**********************************************************************/
/* graphic initialization variables                                    */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code           */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines                     */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /******************************************************************/
  settext();
  /******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
     setfillstyle(SOLID_FILL,BLACK);
     backcolor = BLACK;
     quitcolor = WHITE;
     }
  else {
     setfillstyle(SOLID_FILL,BLUE);
     backcolor = BLUE;
     quitcolor = RED;
     }
  forecolor = WHITE;
  }


/**********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)         {
```

```
    case 'y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'Y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'n': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    case 'N': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/***********************************************************************/
/* This function sets the text default values                          */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/**********************************************************************/
/* main routine calls exer routine                                    */
/**********************************************************************/
void main()
{
   exer();
}
```

```
/*********************************************************************/
/* Routine that asks the question, then depending on the user's answer        */
/* makes necessary explanations                                                 */
/*********************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  6");
  /*****************************************************************/
  outtextxy(2*x,2*y,"Give the postorder listing of the vertices for the following bina-
                    ry tree.");
  /*****************************************************************/
  pieslice(40*x,3*y,0,359,2);
  pieslice(30*x,5*y,0,359,2);
  pieslice(50*x,5*y,0,359,2);
  pieslice(25*x,7*y,0,359,2);
  pieslice(35*x,7*y,0,359,2);
  pieslice(45*x,7*y,0,359,2);
  pieslice(55*x,7*y,0,359,2);
  pieslice(20*x,9*y,0,359,2);
  pieslice(30*x,9*y,0,359,2);
  pieslice(40*x,9*y,0,359,2);
  pieslice(50*x,9*y,0,359,2);
  pieslice(60*x,9*y,0,359,2);
  moveto(20*x,9*y);  lineto(25*x,7*y);
  lineto(30*x,5*y);  lineto(40*x,3*y);
  lineto(50*x,5*y);  lineto(55*x,7*y);
  lineto(60*x,9*y);
  moveto(30*x,9*y);  lineto(35*x,7*y);
  lineto(40*x,9*y);
  moveto(30*x,5*y);  lineto(35*x,7*y);
moveto(50*x,5*y); lineto(45*x,7*y);
lineto(50*x,9*y);
outtextxy(79*x/2,5*y/2,"A");
outtextxy(28*x,5*y,"B");
outtextxy(51*x,5*y,"C");
outtextxy(23*x,7*y,"D");
outtextxy(36*x,7*y,"E");
outtextxy(43*x,7*y,"F");
outtextxy(56*x,7*y,"G");
outtextxy(20*x,19*y/2,"H");
outtextxy(30*x,19*y/2,"I");
outtextxy(40*x,19*y/2,"J");
outtextxy(50*x,19*y/2,"K");
outtextxy(60*x,19*y/2,"L");
/*****************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ():
if(Ch==ESC) confirm_graph_exit();
/*****************************************************************/
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
```

```
switch (Ch)          {
case 'a': outtextxy(47*x,19*y,"a");
  outtextxy(52*x,19*y,"You want to see the algorithm ");
  outtextxy(52*x,20*y,"again. Press any key to continue.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(50*x,37*y/2,179*x/2,21*y);
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  show_alg();
  break;
case 'b': outtextxy(47*x,19*y,"b");
  outtextxy(52*x,19*y,"You want to compare your solu-");
  outtextxy(52*x,20*y,"tion with ours. So press any  ");
  outtextxy(52*x,21*y,"key to see it.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(50*x,37*y/2,179*x/2,22*y);
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  compare_solutions();
  break;
case 'c': outtextxy(47*x,19*y,"c");
  outtextxy(52*x,19*y,"You want to see step by step");
  outtextxy(52*x,20*y,"solution. So press any key to ");
  outtextxy(52*x,21*y,"continue.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(50*x,37*y/2,179*x/2,22*y);
  bar(2*x,13*y,179*x/2,49*y/2);
  setcolor(forecolor);
  step_solution();
  break;
case 'd': outtextxy(47*x,19*y,"d");
  confirm_exit();
  break;
```

```
      default  : break;
     }
  }
  closegraph();
}

/*********************************************************************/
/* This routine gives postorder traversal of a binary tree algorithm          */
/*********************************************************************/
static void show_alg(void)
{
    outtextxy(15*x,14*y,"POSTORDER TRAVERSAL ALGORITHM OF A BINARY
TREE");
    outtextxy(2*x,15*y,"Step 1 (go left)  Go to the left subtree, if one  exists, do a pre-
order");
   outtextxy(2*x,16*y,"      traversal.");
    outtextxy(2*x,17*y,"Step 2 (go right) Go to the right subtree, if one exists, and do
a preorder");
   outtextxy(2*x,18*y,"      traversal.");
   outtextxy(2*x,19*y,"Step 3 (visit)    Visit the root.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}

/*********************************************************************/
/* This routine gives the solution to the exercise to be compared.            */
/*********************************************************************/
static void compare_solutions(void)
{
   setcolor(backcolor);         /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   outtextxy(3*x,29*y/2,"Postorder  Listing");
   moveto(2*x,15*y); lineto(42*x,15*y);
   outtextxy(3*x,16*y,"H, D, I, J, E, B, K, F, L, G, C, A");
   /*****************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);        /* Clean the game field again */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}


/*******************************************************************/
/* This routine gives the step by step solution to the exercise     */
/*******************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);        /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   outtextxy(3*x,29*y/2,"Postorder  Listing");
   moveto(2*x,15*y); lineto(38*x,15*y);
   /*****************************************************************/
   outtextxy(3*x,16*y,"H");    /* Visit  H */
   setcolor(backcolor);
   moveto(20*x,9*y); lineto(25*x,7*y);
   setlinestyle(3,0,3);
   setcolor(forecolor);
   moveto(20*x,9*y); lineto(25*x,7*y);
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,50*x,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   outtextxy(6*x,16*y,"D");          /* Visit D */
   setcolor(backcolor);
   moveto(30*x,5*y); lineto(25*x,7*y);
```

```
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,5*y); lineto(25*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(9*x,16*y,"I");     /* Visit I */
setcolor(backcolor);
moveto(30*x,9*y); lineto(35*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,9*y); lineto(35*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(12*x,16*y,"J");     /* Visit J */
setcolor(backcolor);
moveto(35*x,7*y); lineto(40*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y); lineto(40*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(15*x,16*y,"E");     /* Visit J */
setcolor(backcolor);
moveto(35*x,7*y); lineto(30*x,5*y);
```

```
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y);  lineto(30*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/****************************************************************/
outtextxy(18*x,16*y,"B");          /* Visit B */
setcolor(backcolor);
moveto(40*x,3*y);  lineto(30*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,3*y);  lineto(30*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/****************************************************************/
outtextxy(21*x,16*y,"K");          /* Visit K */
setcolor(backcolor);
moveto(50*x,9*y);  lineto(45*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,9*y);  lineto(45*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/****************************************************************/
outtextxy(24*x,16*y,"F");          /* Visit F */
setcolor(backcolor);
moveto(45*x,7*y);  lineto(50*x,5*y);
```

```
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,7*y); lineto(50*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(27*x,16*y,"L");          /* Visit L */
setcolor(backcolor);
moveto(60*x,9*y); lineto(55*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(60*x,9*y); lineto(55*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(30*x,16*y,"G");          /* Visit G */
setcolor(backcolor);
moveto(50*x,5*y); lineto(55*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,5*y); lineto(55*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(33*x,16*y,"C");          /* Visit C */
setcolor(backcolor);
moveto(50*x,5*y); lineto(40*x,3*y);
```

```
  setlinestyle(3,0,3);
  setcolor(forecolor);
  moveto(50*x,5*y);  lineto(40*x,3*y);
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*****************************************************************/
  outtextxy(36*x,16*y,"A");              /* Visit A */
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*****************************************************************/
  /* Clean the game field  again */
  bar(3*x/2,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  /*****************************************************************/
  /*              Redraw the tree                                  */
  /*****************************************************************/
  moveto(20*x,9*y);  lineto(25*x,7*y);
  lineto(30*x,5*y);  lineto(40*x,3*y);
  lineto(50*x,5*y);  lineto(55*x,7*y);
  lineto(60*x,9*y);
  moveto(30*x,9*y);  lineto(35*x,7*y);
  lineto(40*x,9*y);
  moveto(30*x,5*y);  lineto(35*x,7*y);
  moveto(50*x,5*y);  lineto(45*x,7*y);
  lineto(50*x,9*y);
}
```

```
/***********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch (),
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2.49*y/2);
      setcolor(forecolor);
   }
    switch (ch)         {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;

    case 'n': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    case 'N': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    default : break;
     }
}
```

```
/* PROGRAM   : q457.c
   AUTHOR     : Atilla BALKAN
   DATE        : Apr. 5, 1990
   REVISED    : Apr. 5, 1990

   DESCRIPTION : This program contains the seventh exercise about the
                        binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                                 C compiler Version 2.0.
*/



/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                      /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                       /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)           /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                        /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);
static void example          (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;


/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code      */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines              */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*********************************************************************/
  settext();
  /*********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
      ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
        setfillstyle(SOLID_FILL,BLACK);
        backcolor = BLACK;
        quitcolor = WHITE;
        }
    else {
        setfillstyle(SOLID_FILL,BLUE);
        backcolor = BLUE;
        quitcolor = RED;
        }
   forecolor = WHITE;
  }

/*********************************************************************/
static void confirm_graph_exit(void)
{
    struct _onkey_t *kblist;
    char ch;

    setcolor(backcolor);
    bar(3*x/2,23*y,179*x/2,97*y/4);
    setcolor(quitcolor);
    kblist=chgonkey(NULL);  /* hide any existing hot keys */
    if(_mouse&MS_CURS) mshidecur();
    outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
    ch = getch ();
    while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
        outtextxy(32*x,24*y," Please type y or n");
        ch = getch ();
        if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
        setcolor(backcolor);
        bar(31*x,23*y,69*x,97*y/4);
        setcolor(quitcolor);
     }
     switch (ch)          {
      case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
}




/**********************************************************************/
/* This function sets the text default values                         */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen             */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
   }

/***********************************************************************/
/* main routine that calls exer routine                                */
/***********************************************************************/
void main()
{
  exer();
}

/***********************************************************************/
/* Routine that asks the question, then depending on the user's answer */
/* makes necessary explanations                                        */
/***********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  7");
   /*******************************************************************/
   outtextxy(2*x,2*y,"Give the postorder listing of the vertices for the following bina-
                        ry tree.");
   /*******************************************************************/
```

```
pieslice(40*x,3*y,0,359,2);
pieslice(35*x,5*y,0,359,2);
pieslice(45*x,5*y,0,359,2);
pieslice(30*x,7*y,0,359,2);
pieslice(38*x,7*y,0,359,2);
pieslice(50*x,7*y,0,359,2);
pieslice(25*x,9*y,0,359,2);
pieslice(35*x,9*y,0,359,2);
pieslice(42*x,9*y,0,359,2);
pieslice(46*x,9*y,0,359,2);
pieslice(55*x,9*y,0,359,2);
pieslice(20*x,11*y,0,359,2);
pieslice(30*x,11*y,0,359,2);
pieslice(38*x,11*y,0,359,2);
pieslice(42*x,11*y,0,359,2);
pieslice(50*x,11*y,0,359,2);
pieslice(60*x,11*y,0,359,2);
moveto(20*x,11*y); lineto(25*x,9*y);
lineto(30*x,7*y);  lineto(35*x,5*y);
lineto(40*x,3*y);  lineto(45*x,5*y);
lineto(50*x,7*y);  lineto(55*x,9*y);
lineto(60*x,11*y);
moveto(30*x,11*y); lineto(35*x,9*y);
lineto(38*x,7*y);  lineto(42*x,9*y);
lineto(38*x,11*y);
moveto(35*x,5*y); lineto(38*x,7*y);
moveto(42*x,11*y); lineto(46*x,9*y);
lineto(50*x,11*y);
moveto(46*x,9*y); lineto(50*x,7*y);
outtextxy(79*x/2,5*y/2,"A");
outtextxy(33*x,5*y,"B");
outtextxy(46*x,5*y,"C");
outtextxy(28*x,7*y,"D");
outtextxy(39*x,7*y,"E");
outtextxy(51*x,7*y,"F");
outtextxy(23*x,9*y,"G");
outtextxy(33*x,9*y,"H");
outtextxy(43*x,9*y,"I");
outtextxy(47*x,9*y,"J");
outtextxy(56*x,9*y,"K");
outtextxy(20*x,23*y/2,"L");
outtextxy(30*x,23*y/2,"M");
outtextxy(38*x,23*y/2,"N");
outtextxy(42*x,23*y/2,"O");
outtextxy(50*x,23*y/2,"P");
outtextxy(60*x,23*y/2,"Q");
/*******************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
    switch (Ch)        {
    case 'a': outtextxy(47*x,19*y,"a");
      outtextxy(52*x,19*y,"You want to see the algorithm ");
      outtextxy(52*x,20*y,"again. Press any key to continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
```

```
        bar(50*x,37*y/2,179*x/2,21*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        show_alg();
        break;
      case 'b': outtextxy(47*x,19*y,"b");
        outtextxy(52*x,19*y,"You want to compare your solu-");
        outtextxy(52*x,20*y,"tion with ours. So press any  ");
        outtextxy(52*x,21*y,"key to see it.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        compare_solutions();
        break;
      case 'c': outtextxy(47*x,19*y,"c");
        outtextxy(52*x,19*y,"You want to see step by step");
        outtextxy(52*x,20*y,"solution. So press any key to ");
        outtextxy(52*x,21*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
    }
  }
  closegraph();
}
```

```
/***********************************************************************/
/* This routine gives postorder traversal of a binary tree algorithm           */
/***********************************************************************/
static void show_alg(void)
{
  outtextxy(15*x,14*y,"POSTORDER TRAVERSAL ALGORITHM OF A BINARY
                        TREE");
  outtextxy(2*x,15*y,"Step 1 (go left)  Go to the left subtree, if one  exists, do a pre-
                        order");
  outtextxy(2*x,16*y,"      traversal.");
  outtextxy(2*x,17*y,"Step 2 (go right) Go to the right subtree, if one exists, and do
                        a preorder");
  outtextxy(2*x,18*y,"      traversal.");
  outtextxy(2*x,19*y,"Step 3 (visit)    Visit the root.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}
/***********************************************************************/
/* This routine gives the solution to the exercise to be compared.             */
/***********************************************************************/
static void compare_solutions(void)
{

  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  outtextxy(3*x,29*y/2,"Postorder  Listing");
  moveto(2*x,15*y); lineto(60*x,15*y);
  outtextxy(3*x,16*y,"L, G, D, M, H, N, I, E, B, O, P, J, Q, K, F, C, A");
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field again */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/************************************************************************/
/* This routine gives the step by step solution to the exercise        */
/************************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   outtextxy(3*x,29*y/2,"Postorder  Listing");
   moveto(2*x,15*y); lineto(53*x,15*y);
   /*********************************************************************/
   outtextxy(3*x,16*y,"L");     /* Visit L */
   setcolor(backcolor);
   moveto(20*x,11*y); lineto(25*x,9*y);
   setlinestyle(3,0,3);
   setcolor(forecolor);
   moveto(20*x,11*y); lineto(25*x,9*y);
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,50*x,49*y/2);
   setcolor(forecolor);
   /*********************************************************************/
   outtextxy(6*x,16*y,"G");     /* Visit G */
   setcolor(backcolor);
   moveto(30*x,7*y); lineto(25*x,9*y);
   setlinestyle(3,0,3);
   setcolor(forecolor);
   moveto(30*x,7*y); lineto(25*x,9*y);
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,50*x,49*y/2);
   setcolor(forecolor);
   /*********************************************************************/
   outtextxy(9*x,16*y,"D");     /* Visit D */
```

```
setcolor(backcolor);
moveto(30*x,7*y); lineto(35*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,7*y); lineto(35*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(12*x,16*y,"M");     /* Visit M */
setcolor(backcolor);
moveto(30*x,11*y); lineto(35*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,11*y); lineto(35*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(15*x,16*y,"H");     /* Visit H */
setcolor(backcolor);
moveto(38*x,7*y); lineto(35*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,7*y); lineto(35*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(18*x,16*y,"N");     /* Visit N */
```

```
setcolor(backcolor);
moveto(38*x,11*y); lineto(42*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,11*y); lineto(42*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(21*x,16*y,"I");     /* Visit I */
setcolor(backcolor);
moveto(38*x,7*y); lineto(42*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,7*y); lineto(42*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(24*x,16*y,"E");     /* Visit E */
setcolor(backcolor);
moveto(38*x,7*y); lineto(35*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,7*y); lineto(35*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(27*x,16*y,"B");     /* Visit B */
```

```
setcolor(backcolor);
moveto(40*x,3*y); lineto(35*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,3*y); lineto(35*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(30*x,16*y,"O");     /* Visit O */
setcolor(backcolor);
moveto(42*x,11*y); lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(42*x,11*y); lineto(46*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(33*x,16*y,"P");     /* Visit P */
setcolor(backcolor);
moveto(50*x,11*y); lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,11*y); lineto(46*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(36*x,16*y,"J");     /* Visit J */
```

```
setcolor(backcolor);
moveto(50*x,7*y); lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,7*y); lineto(46*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(39*x,16*y,"Q");     /* Visit Q */
setcolor(backcolor);
moveto(55*x,9*y); lineto(60*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,9*y); lineto(60*x,11*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(42*x,16*y,"K");     /* Visit K */
setcolor(backcolor);
moveto(55*x,9*y); lineto(50*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,9*y); lineto(50*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(45*x,16*y,"F");     /* Visit F */
```

```
setcolor(backcolor);
moveto(45*x,5*y); lineto(50*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,5*y); lineto(50*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(48*x,16*y,"C");     /* Visit C */
setcolor(backcolor);
moveto(45*x,5*y); lineto(40*x,3*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,5*y); lineto(40*x,3*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(51*x,16*y,"A");     /* Visit A */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* Clean the game field  again */
bar(3*x/2,47*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/*              Redraw the tree                                   */
/*****************************************************************/
moveto(20*x,11*y); lineto(25*x,9*y);
```

```
   lineto(30*x,7*y);  lineto(35*x,5*y);
   lineto(40*x,3*y);  lineto(45*x,5*y);
   lineto(50*x,7*y);  lineto(55*x,9*y);
   lineto(60*x,11*y);
   moveto(30*x,11*y); lineto(35*x,9*y);
   lineto(38*x,7*y);  lineto(42*x,9*y);
   lineto(38*x,11*y);
   moveto(35*x,5*y); lineto(38*x,7*y);
   moveto(42*x,11*y); lineto(46*x,9*y);
   lineto(50*x,11*y);
   moveto(46*x,9*y); lineto(50*x,7*y);
}

/*********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
   switch (ch)        {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;
```

```
        case 'n': setcolor(backcolor);
              bar(46*x,37*y/2,179*x/2,22*y);
              setcolor(forecolor);
              break;

        case 'N': setcolor(backcolor);
              bar(46*x,37*y/2,179*x/2,22*y);
              setcolor(forecolor);
              break;

        default : break;
        }
}
```

```
/* PROGRAM    : q458.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 4, 1990
   REVISED    : Apr. 4, 1990

   DESCRIPTION : This program contains the eighth exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;


/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code          */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines                   */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/***********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)         {
    case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
      case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
      case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}




/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/********************************************************************/
/* Equivalent of press_a_key function for graphics screen          */
/********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/********************************************************************/
/* main routine that calls exer routine                             */
/********************************************************************/
void main()
{
  exer();
}

/********************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                     */
/********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  8");
   /*****************************************************************/
   outtextxy(2*x,2*y,"Give the inorder listing of the vertices for the following binary
                         tree.");
   /*****************************************************************/
```

```
pieslice(40*x,3*y,0,359,2);
pieslice(30*x,5*y,0,359,2);
pieslice(50*x,5*y,0,359,2);
pieslice(25*x,7*y,0,359,2);
pieslice(35*x,7*y,0,359,2);
pieslice(45*x,7*y,0,359,2);
pieslice(55*x,7*y,0,359,2);
pieslice(20*x,9*y,0,359,2);
pieslice(30*x,9*y,0,359,2);
pieslice(40*x,9*y,0,359,2);
pieslice(50*x,9*y,0,359,2);
pieslice(60*x,9*y,0,359,2);
moveto(20*x,9*y); lineto(25*x,7*y);
lineto(30*x,5*y); lineto(40*x,3*y);
lineto(50*x,5*y); lineto(55*x,7*y);
lineto(60*x,9*y);
moveto(30*x,9*y); lineto(35*x,7*y);
lineto(40*x,9*y);
moveto(30*x,5*y); lineto(35*x,7*y);
moveto(50*x,5*y); lineto(45*x,7*y);
lineto(50*x,9*y);
outtextxy(79*x/2,5*y/2,"A");
outtextxy(28*x,5*y,"B");
outtextxy(51*x,5*y,"C");
outtextxy(23*x,7*y,"D");
outtextxy(36*x,7*y,"E");
outtextxy(43*x,7*y,"F");
outtextxy(56*x,7*y,"G");
outtextxy(20*x,19*y/2,"H");
outtextxy(30*x,19*y/2,"I");
outtextxy(40*x,19*y/2,"J");
outtextxy(50*x,19*y/2,"K");
outtextxy(60*x,19*y/2,"L");
/*********************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"     a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"     b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"     c) I want to see step by step solution.");
outtextxy(15*x,18*y,"     d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
 while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,19*y,"   Please type a, b, c or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
   setcolor(backcolor);
   bar(50*x,37*y/2,88*x,20*y);
   setcolor(forecolor);
   }
  }
   switch (Ch)        {
   case 'a': outtextxy(47*x,19*y,"a");
     outtextxy(52*x,19*y,"You want to see the algorithm ");
     outtextxy(52*x,20*y,"again. Press any key to continue.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,37*y/2,179*x/2,21*y);
     bar(2*x,13*y,179*x/2,49*y/2);
     setcolor(forecolor);
     show_alg();
     break;
  case 'b': outtextxy(47*x,19*y,"b");
     outtextxy(52*x,19*y,"You want to compare your solu-");
     outtextxy(52*x,20*y,"tion with ours. So press any  ");
     outtextxy(52*x,21*y,"key to see it.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,37*y/2,179*x/2,22*y);
     bar(2*x,13*y,179*x/2,49*y/2);
```

```
            setcolor(forecolor);
            compare_solutions();
            break;
          case 'c': outtextxy(47*x,19*y,"c");
            outtextxy(52*x,19*y,"You want to see step by step");
            outtextxy(52*x,20*y,"solution. So press any key to ");
            outtextxy(52*x,21*y,"continue.");
            Pause(30*x,24*y);
            setcolor(backcolor);
            bar(50*x,37*y/2,179*x/2,22*y);
            bar(2*x,13*y,179*x/2,49*y/2);
            setcolor(forecolor);
            step_solution();
            break;
          case 'd': outtextxy(47*x,19*y,"d");
            confirm_exit();
            break;
          default  : break;
        }
    }
    closegraph();
}

/*********************************************************************/
/* This routine gives inorder traversal of a binary tree algorithm         */
/*********************************************************************/
static void show_alg(void)
{
    outtextxy(15*x,14*y,"INORDER TRAVERSAL ALGORITHM OF A BINARY
                        TREE");
    outtextxy(2*x,15*y,"Step 1 (go left)  Go to the left subtree, if one  exists, do a in-
                        order");
    outtextxy(2*x,16*y,"       traversal.");
    outtextxy(2*x,17*y,"Step 2 (visit)    Visit the root.");
    outtextxy(2*x,18*y,"Step 3 (go right) Go to the right subtree, if one exists, and do
                        a inorder");
```

```
   outtextxy(2*x,19*y,"      traversal.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}

/********************************************************************/
/* This routine gives the solution to the exercise to be compared.          */
/********************************************************************/
static void compare_solutions(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   outtextxy(3*x,29*y/2,"Inorder  Listing");
   moveto(2*x,15*y);  lineto(42*x,15*y);
   outtextxy(3*x,16*y,"H, D, B, I, E, J, A, F, K, C, G, L");
   /*****************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field again */
   bar(2*x,47*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}
```

```
/*******************************************************************/
/* This routine gives the step by step solution to the exercise     */
/*******************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  outtextxy(3*x,29*y/2,"Inorder  Listing");
  moveto(2*x,15*y);  lineto(38*x,15*y);
  /****************************************************************/
  outtextxy(3*x,16*y,"H");     /* Visit  H */
  setcolor(backcolor);
  moveto(20*x,9*y);  lineto(25*x,7*y);
  setlinestyle(3,0,3);
  setcolor(forecolor);
  moveto(20*x,9*y);  lineto(25*x,7*y);
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /****************************************************************/
  outtextxy(6*x,16*y,"D");           /* Visit D */
  setcolor(backcolor);
  moveto(30*x,5*y);  lineto(25*x,7*y);
  setlinestyle(3,0,3);
  setcolor(forecolor);
  moveto(30*x,5*y);  lineto(25*x,7*y);
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /****************************************************************/
  outtextxy(9*x,16*y,"B");     /* Visit  B */
```

```
setcolor(backcolor);
moveto(30*x,5*y); lineto(35*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,5*y); lineto(35*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(12*x,16*y,"I");     /* Visit I */
setcolor(backcolor);
moveto(35*x,7*y); lineto(30*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y); lineto(30*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(15*x,16*y,"E");     /* Visit E */
setcolor(backcolor);
moveto(35*x,7*y); lineto(40*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,7*y); lineto(40*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(18*x,16*y,"J");          /* Visit J */
```

```
setcolor(backcolor);
moveto(40*x,3*y); lineto(30*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(40*x,3*y); lineto(30*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(21*x,16*y,"A");          /* Visit A */
setcolor(backcolor);
moveto(50*x,5*y); lineto(40*x,3*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,5*y); lineto(40*x,3*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(24*x,16*y,"F");          /* Visit F */
setcolor(backcolor);
moveto(45*x,7*y); lineto(50*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,7*y); lineto(50*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(27*x,16*y,"K");          /* Visit K */
```

```
setcolor(backcolor);
moveto(50*x,5*y);  lineto(45*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,5*y);  lineto(45*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/******************************************************************/
outtextxy(30*x,16*y,"C");          /* Visit C */
setcolor(backcolor);
moveto(50*x,5*y);  lineto(55*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,5*y);  lineto(55*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/******************************************************************/
outtextxy(33*x,16*y,"G");          /* Visit G */
setcolor(backcolor);
moveto(55*x,7*y);  lineto(60*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,7*y);  lineto(60*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/******************************************************************/
outtextxy(36*x,16*y,"L");          /* Visit L */
```

```
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /********************************************************************/
    /* Clean the game field  again */
    bar(3*x/2,47*y/4,179*x/2,49*y/2);
    setcolor(forecolor);
    /********************************************************************/
    /*              Redraw the tree                                      */
    /********************************************************************/
    moveto(20*x,9*y);  lineto(25*x,7*y);
    lineto(30*x,5*y);  lineto(40*x,3*y);
    lineto(50*x,5*y);  lineto(55*x,7*y);
    lineto(60*x,9*y);
    moveto(30*x,9*y);  lineto(35*x,7*y);
    lineto(40*x,9*y);
    moveto(30*x,5*y);  lineto(35*x,7*y);
    moveto(50*x,5*y);  lineto(45*x,7*y);
    lineto(50*x,9*y);
}
```

```
/*******************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,23*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,22*y,179*x/2,49*y/2);
       setcolor(forecolor);
    }
    switch (ch)          {
     case 'y': in_the_exercise = 0;
           break;
     case 'Y': in_the_exercise = 0;
           break;

     case 'n': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

     case 'N': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

     default : break;
     }
}
```

```
/* PROGRAM     : q459.c
   AUTHOR      : Atilla BAKAN
   DATE        : Apr. 5, 1990
   REVISED     : Apr. 5, 1990

   DESCRIPTION : This program contains the ninth exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/
/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif

#define  GRAPH_T_DEFINED
```

```
/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);
static void example          (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);


/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
 int in_the_exercise = 1;



/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/********************************************************************/
/* This function is used for including drivers to the executable code          */
/********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/********************************************************************/
/* This fuction initializes the necessary graphical routines                    */
/********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*******************************************************************/
  settext();
  /*******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/*********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}




/************************************************************************/
/* This function sets the text default values                         */
/************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen          */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine  calls exer routine                                  */
/*********************************************************************/
void main()
{
  exer();
}

/*********************************************************************/
/* Routine that asks the question, then depending on the user's answer     */
/* makes necessary explanations                                      */
/*********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  9");
   /*****************************************************************/
   outtextxy(2*x,2*y,"Give the inorder listing of the vertices for the following binary
                    tree.");
   /*****************************************************************/
```

```
pieslice(40*x,3*y,0,359,2);
pieslice(35*x,5*y,0,359,2);
pieslice(45*x,5*y,0,359,2);
pieslice(30*x,7*y,0,359,2);
pieslice(38*x,7*y,0,359,2);
pieslice(50*x,7*y,0,359,2);
pieslice(25*x,9*y,0,359,2);
pieslice(35*x,9*y,0,359,2);
pieslice(42*x,9*y,0,359,2);
pieslice(46*x,9*y,0,359,2);
pieslice(55*x,9*y,0,359,2);
pieslice(20*x,11*y,0,359,2);
pieslice(30*x,11*y,0,359,2);
pieslice(38*x,11*y,0,359,2);
pieslice(42*x,11*y,0,359,2);
pieslice(50*x,11*y,0,359,2);
pieslice(60*x,11*y,0,359,2);
moveto(20*x,11*y); lineto(25*x,9*y);
lineto(30*x,7*y);  lineto(35*x,5*y);
lineto(40*x,3*y);  lineto(45*x,5*y);
lineto(50*x,7*y);  lineto(55*x,9*y);
lineto(60*x,11*y);
moveto(30*x,11*y); lineto(35*x,9*y);
lineto(38*x,7*y);  lineto(42*x,9*y);
lineto(38*x,11*y);
moveto(35*x,5*y);  lineto(38*x,7*y);
moveto(42*x,11*y); lineto(46*x,9*y);
lineto(50*x,11*y);
moveto(46*x,9*y); lineto(50*x,7*y);
outtextxy(79*x/2,5*y/2,"A");
outtextxy(33*x,5*y,"B");
outtextxy(46*x,5*y,"C");
outtextxy(28*x,7*y,"D");
outtextxy(39*x,7*y,"E");
outtextxy(51*x,7*y,"F");
outtextxy(23*x,9*y,"G");
outtextxy(33*x,9*y,"H");
outtextxy(43*x,9*y,"I");
outtextxy(47*x,9*y,"J");
outtextxy(56*x,9*y,"K");
outtextxy(20*x,23*y/2,"L");
outtextxy(30*x,23*y/2,"M");
outtextxy(38*x,23*y/2,"N");
outtextxy(42*x,23*y/2,"O");
outtextxy(50*x,23*y/2,"P");
outtextxy(60*x,23*y/2,"Q");
/*****************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
/*****************************************************************/
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
    switch (Ch)          {
    case 'a': outtextxy(47*x,19*y,"a");
      outtextxy(52*x,19*y,"You want to see the algorithm ");
      outtextxy(52*x,20*y," again. Press any key to continue.");
      Pause(30*x,24*y);
```

```
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,21*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        show_alg();
        break;
      case 'b': outtextxy(47*x,19*y,"b");
        outtextxy(52*x,19*y,"You want to compare your solu-");
        outtextxy(52*x,20*y,"tion with ours. So press any  ");
        outtextxy(52*x,21*y,"key to see it.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        compare_solutions();
        break;
      case 'c': outtextxy(47*x,19*y,"c");
        outtextxy(52*x,19*y,"You want to see step by step");
        outtextxy(52*x,20*y,"solution. So press any key to ");
        outtextxy(52*x,21*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
     }
  }
  closegraph();
}
```

```
/**********************************************************************/
/* This routine gives inorder traversal of a binary tree algorithm          */
/**********************************************************************/
static void show_alg(void)
{
  outtextxy(15*x,14*y,"INORDER TRAVERSAL ALGORITHM OF A BINARY
                        TREE");
  outtextxy(2*x,15*y,"Step 1 (go left)  Go to the left subtree, if one  exists, do a pre-
                        order");
  outtextxy(2*x,16*y,"      traversal.");
  outtextxy(2*x,17*y,"Step 2 (visit)    Visit the root.");
  outtextxy(2*x,18*y,"Step 3 (go right) Go to the right subtree, if one exists, and do
                        a preorder");
  outtextxy(2*x,19*y,"      traversal.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}
/**********************************************************************/
/* This routine gives the solution to the exercise to be compared.           */
/**********************************************************************/
static void compare_solutions(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  /********************************************************************/
  outtextxy(3*x,29*y/2,"Inorder  Listing");
  moveto(2*x,15*y);  lineto(60*x,15*y);
  outtextxy(3*x,16*y,"L, G, D, B, M, H, E, N, I, A, C, O, J, P, F, K, Q");
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field again */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/************************************************************************/
/* This routine gives the step by step solution to the exercise           */
/************************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);        /* Clean the game field */
  bar(2*x,47*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  outtextxy(3*x,29*y/2,"Inorder  Listing");
  moveto(2*x,15*y);  lineto(53*x,15*y);
  /*********************************************************************/
  outtextxy(3*x,16*y,"L");     /* Visit L */
  setcolor(backcolor);
  moveto(20*x,11*y);  lineto(25*x,9*y);
  setlinestyle(3,0,3);
  setcolor(forecolor);
  moveto(20*x,11*y);  lineto(25*x,9*y);
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  outtextxy(6*x,16*y,"G");     /* Visit G */
  setcolor(backcolor);
  moveto(30*x,7*y);  lineto(25*x,9*y);
  setlinestyle(3,0,3);
  setcolor(forecolor);
  moveto(30*x,7*y);  lineto(25*x,9*y);
  setlinestyle(0,0,3);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  outtextxy(9*x,16*y,"D");     /* Visit D */
```

```
setcolor(backcolor);
moveto(30*x,7*y); lineto(35*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,7*y); lineto(35*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************************/
outtextxy(12*x,16*y,"B");     /* Visit B */
setcolor(backcolor);
moveto(38*x,7*y); lineto(35*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,7*y); lineto(35*x,5*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************************/
outtextxy(15*x,16*y,"M");     /* Visit M */
setcolor(backcolor);
moveto(30*x,11*y); lineto(35*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(30*x,11*y); lineto(35*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************************/
outtextxy(18*x,16*y,"H");     /* Visit H */
```

```
setcolor(backcolor);
moveto(38*x,7*y);  lineto(35*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,7*y);  lineto(35*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(21*x,16*y,"E");     /* Visit E */
setcolor(backcolor);
moveto(38*x,7*y);  lineto(42*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,7*y);  lineto(42*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(24*x,16*y,"N");     /* Visit N */
setcolor(backcolor);
moveto(38*x,11*y);  lineto(42*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(38*x,11*y);  lineto(42*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(27*x,16*y,"I");     /* Visit I */
```

```
setcolor(backcolor);
moveto(35*x,5*y); lineto(40*x,3*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,5*y); lineto(40*x,3*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(30*x,16*y,"A");     /* Visit A */
setcolor(backcolor);
moveto(45*x,5*y); lineto(40*x,3*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,5*y); lineto(40*x,3*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(33*x,16*y,"C");     /* Visit C */
setcolor(backcolor);
moveto(45*x,5*y); lineto(50*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,5*y); lineto(50*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(36*x,16*y,"O");     /* Visit O */
```

```
setcolor(backcolor);
moveto(42*x,11*y); lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(42*x,11*y); lineto(46*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(39*x,16*y,"J");     /* Visit J */
setcolor(backcolor);
moveto(50*x,11*y); lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,11*y); lineto(46*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(42*x,16*y,"P");     /* Visit P */
setcolor(backcolor);
moveto(50*x,7*y); lineto(46*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(50*x,7*y); lineto(46*x,9*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(45*x,16*y,"F");     /* Visit F */
```

```
setcolor(backcolor);
moveto(55*x,9*y); lineto(50*x,7*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,9*y); lineto(50*x,7*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(48*x,16*y,"K");     /* Visit K */
setcolor(backcolor);
moveto(55*x,9*y); lineto(60*x,11*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,9*y); lineto(60*x,11*y);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(51*x,16*y,"Q");     /* Visit Q */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/* Clean the game field  again */
bar(3*x/2,47*y/4,179*x/2,49*y/2);
setcolor(forecolor);
/*****************************************************************/
/*              Redraw the tree                                   */
/*****************************************************************/
moveto(20*x,11*y); lineto(25*x,9*y);
```

```
   lineto(30*x,7*y);  lineto(35*x,5*y);
   lineto(40*x,3*y);  lineto(45*x,5*y);
   lineto(50*x,7*y);  lineto(55*x,9*y);
   lineto(60*x,11*y);
   moveto(30*x,11*y); lineto(35*x,9*y);
   lineto(38*x,7*y);  lineto(42*x,9*y);
   lineto(38*x,11*y);
   moveto(35*x,5*y); lineto(38*x,7*y);
   moveto(42*x,11*y); lineto(46*x,9*y);
   lineto(50*x,11*y);
   moveto(46*x,9*y); lineto(50*x,7*y);
}

/*********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
   switch (ch)         {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;
    case 'n': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    case 'N': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    default : break;
    }
}
```

```
/* PROGRAM   : q4510.c
   AUTHOR     : Atilla BAKAN
   DATE        : Mar. 22, 1990
   REVISED   : Apr. 17, 1990

   DESCRIPTION : This program contains the tenth exercise about the
                         binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                                  C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example          (void);
static void show_alg         (void);
static void step_solution     (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/***********************************************************************/
/* miscellaneous global variables                                      */
/***********************************************************************/
 int in_the_exercise = 1;


/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*******************************************************************/
/* This function is used for including drivers to the executable code          */
/*******************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*******************************************************************/
/* This fuction initializes the necessary graphical routines                  */
/*******************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*****************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*****************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*****************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*****************************************************************/
  settext();
  /*****************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
   ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
     setfillstyle(SOLID_FILL,BLACK);
     backcolor = BLACK;
     quitcolor = WHITE;
     }
  else {
     setfillstyle(SOLID_FILL,BLUE);
     backcolor = BLUE;
     quitcolor = RED;
     }
  forecolor = WHITE;
 }

/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}




/***********************************************************************/
/* This function sets the text default values                          */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
   }

/***********************************************************************/
/* main routine that calls exer routine                                */
/***********************************************************************/
void main()
{
   exer();
}

/***********************************************************************/
/* Routine that asks the question, then depending on the user's answer */
/* makes necessary explanations                                        */
/***********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE 10");
   /*******************************************************************/
   outtextxy(15*x,2*y,"Evaluate the following Polish notation expression");
   outtextxy(30*x,3*y,"+ * + 3 4 - 1 2 - 3 / 4 2");
   /*******************************************************************/
while (in_the_exercise == 1) {
 outtextxy(15*x,14*y,"Choose one of the following, as you need :");
 outtextxy(15*x,15*y,"     a) I'm done, I want to compare my solution with yours.");
 outtextxy(15*x,16*y,"     b) I want to see step by step solution.");
 outtextxy(15*x,17*y,"     c) This is enough for me, I want to exit.");
 outtextxy(15*x,18*y,"Enter your choice here --->");
 Ch = getch ();
 if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,18*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
   }
    switch (Ch)          {
   case 'a': outtextxy(47*x,18*y,"a");
     outtextxy(52*x,18*y,"You want to compare your solu-");
     outtextxy(52*x,19*y,"tion with ours. So press any  ");
     outtextxy(52*x,20*y,"key to see it.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,35*y/2,179*x/2,22*y);
     bar(2*x,4*y,179*x/2,49*y/2);
     setcolor(forecolor);
     compare_solutions();
     break;
   case 'b': outtextxy(47*x,18*y,"b");
     outtextxy(52*x,18*y,"You want to see step by step");
     outtextxy(52*x,19*y,"solution. So press any key to ");
     outtextxy(52*x,20*y,"continue.");
     Pause(30*x,24*y);
     setcolor(backcolor);
```

```
      bar(50*x,35*y/2,179*x/2,22*y);
      bar(2*x,4*y,179*x/2,49*y/2);
      setcolor(forecolor);
      step_solution();
      break;
    case 'c': outtextxy(47*x,18*y,"c");
      confirm_exit();
      break;
    default  : break;
   }
 }
  closegraph();
}
```

```
/***********************************************************************/
/* This routine gives the solution to the exercise to be compared.     */
/***********************************************************************/
static void compare_solutions(void)
{

   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
   outtextxy(35*x,10*y,"The answer is  -6");
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field  again */
   bar(2*x,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
}
```

```
/*************************************************************************/
/* This routine gives the step by step solution to the exercise          */
/*************************************************************************/
static void step_solution(void)
{

  setcolor(backcolor);        /* Clean the game field */
  bar(3*x/2,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  outtextxy(30*x,8*y,"  + * + 3 4 - 1 2 - 3 / 4 2");
  Pause(30*x,24*y);
  /*********************************************************************/
  outtextxy(30*x,9*y,"=  + * 7 - 1 2 - 3 / 4 2");
  Pause(30*x,24*y);
  /*********************************************************************/
  outtextxy(30*x,10*y,"=  + * 7 (-1) - 3 / 4 2");
  Pause(30*x,24*y);
  /*********************************************************************/
  outtextxy(30*x,11*y,"=  + (-7) - 3 / 4 2");
  Pause(30*x,24*y);
  /*********************************************************************/
  outtextxy(30*x,12*y,"=  + (-7) - 3 2");
  Pause(30*x,24*y);
  /*********************************************************************/
  outtextxy(30*x,13*y,"=  + (-7) 1");
  Pause(30*x,24*y);
  /*********************************************************************/
  outtextxy(30*x,14*y,"=  -6");
  Pause(30*x,24*y);
  /*********************************************************************/
  setcolor(backcolor);        /* Clean the game field  again */
  bar(2*x,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/******************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,22*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,21*y,179*x/2,49*y/2);
       setcolor(forecolor);
    }
    switch (ch)          {
     case 'y': in_the_exercise = 0;
           break;
     case 'Y': in_the_exercise = 0;
           break;

     case 'n': setcolor(backcolor);
           bar(46*x,35*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

     case 'N': setcolor(backcolor);
           bar(46*x,35*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

     default : break;
     }
}
```

```
/* PROGRAM   : q4511.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 5, 1990
   REVISED   : Apr. 5, 1990

   DESCRIPTION : This program contains the eleventh exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example          (void);
static void show_alg         (void);
static void step_solution     (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
 int in_the_exercise = 1;


/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/**********************************************************************/
/* This function is used for including drivers to the executable code        */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/**********************************************************************/
/* This fuction initializes the necessary graphical routines              */
/**********************************************************************/
static void init_graph(void)
{
  int xasp. yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*********************************************************************/
  settext();
  /*********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
      ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
        setfillstyle(SOLID_FILL,BLACK);
        backcolor = BLACK;
        quitcolor = WHITE;
        }
    else {
        setfillstyle(SOLID_FILL,BLUE);
        backcolor = BLUE;
        quitcolor = RED;
        }
   forecolor = WHITE;
  }

/*********************************************************************/
static void confirm_graph_exit(void)
{
    struct _onkey_t *kblist;
    char ch;

    setcolor(backcolor);
    bar(3*x/2,23*y,179*x/2,97*y/4);
    setcolor(quitcolor);
    kblist=chgonkey(NULL);  /* hide any existing hot keys */
    if(_mouse&MS_CURS) mshidecur();
    outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
    ch = getch ();
    while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
        outtextxy(32*x,24*y," Please type y or n");
        ch = getch ();
        if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N' ))
        setcolor(backcolor);
        bar(31*x,23*y,69*x,97*y/4);
        setcolor(quitcolor);
     }
     switch (ch)          {
      case 'y': closegraph();
```

```
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   default : break;
   }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);    /* restore any hidden hot keys */
}



/**********************************************************************/
/* This function sets the text default values                        */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/**********************************************************************/
/* main routine that calls exer routine                              */
/**********************************************************************/
void main()
{
  exer();
}

/**********************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                       */
/**********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  11");
   /*******************************************************************/
   outtextxy(15*x,2*y,"Evaluate the following Polish notation expression");
   outtextxy(30*x,3*y,"+ * 4 / 6 2 - + 4 2 5");
   /*******************************************************************/
while (in_the_exercise == 1) {
 outtextxy(15*x,14*y,"Choose one of the following, as you need :");
 outtextxy(15*x,15*y,"    a) I'm done, I want to compare my solution with yours.");
 outtextxy(15*x,16*y,"    b) I want to see step by step solution.");
 outtextxy(15*x,17*y,"    c) This is enough for me, I want to exit.");
 outtextxy(15*x,18*y,"Enter your choice here --->");
 Ch = getch ();
 if(Ch==ESC) confirm_graph_exit();
   while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
     outtextxy(48*x,18*y,"    Please type a, b, c or d");
     Ch = getch ();
     if(Ch==ESC) confirm_graph_exit();
     if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
     setcolor(backcolor);
     bar(50*x,35*y/2,88*x,20*y);
     setcolor(forecolor);
     }
   }
     switch (Ch)        {
    case 'a': outtextxy(47*x,18*y,"a");
      outtextxy(52*x,18*y,"You want to compare your solu-");
      outtextxy(52*x,19*y,"tion with ours. So press any  ");
      outtextxy(52*x,20*y,"key to see it.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,35*y/2,179*x/2,22*y);
      bar(2*x,4*y,179*x/2,49*y/2);
      setcolor(forecolor);
      compare_solutions();
      break;
    case 'b': outtextxy(47*x,18*y,"b");
      outtextxy(52*x,18*y,"You want to see step by step");
      outtextxy(52*x,19*y,"solution. So press any key to ");
      outtextxy(52*x,20*y,"continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
```

```
        bar(50*x,35*y/2,179*x/2,22*y);
        bar(2*x,4*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'c': outtextxy(47*x,18*y,"c");
        confirm_exit();
        break;
      default  : break;
     }
  }
  closegraph();
}
```

```
/**********************************************************************/
/* This routine gives the solution to the exercise to be compared.    */
/**********************************************************************/
static void compare_solutions(void)
{

  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
  outtextxy(35*x,10*y,"The answer is  13");
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field  again */
  bar(2*x,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/************************************************************************/
/* This routine gives the step by step solution to the exercise          */
/************************************************************************/
static void step_solution(void)
{

   setcolor(backcolor);          /* Clean the game field */
   bar(3*x/2,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
   /*********************************************************************/
   outtextxy(30*x,8*y,"  + * 4 / 6 2 - + 4 2 5");
   Pause(30*x,24*y);
   /*********************************************************************/
   outtextxy(30*x,9*y,"= + * 4 3 - + 4 2 5");
   Pause(30*x,24*y);
   /*********************************************************************/
   outtextxy(30*x,10*y,"= + 12 - + 4 2 5");
   Pause(30*x,24*y);
   /*********************************************************************/
   outtextxy(30*x,11*y,"= + 12 - 6 5");
   Pause(30*x,24*y);
   /*********************************************************************/
   outtextxy(30*x,12*y,"= + 12 1");
   Pause(30*x,24*y);
   /*********************************************************************/
   outtextxy(30*x,13*y,"= 13");
   Pause(30*x,24*y);
   /*********************************************************************/
   setcolor(backcolor);          /* Clean the game field  again */
   bar(2*x,4*y,179*x/2,49*y/2);
   setcolor(forecolor);
}
```

```
/***************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,22*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,21*y,179*x/2,49*y/2);
       setcolor(forecolor);
    }
    switch (ch)         {
     case 'y': in_the_exercise = 0;
            break;
     case 'Y': in_the_exercise = 0;
            break;

     case 'n': setcolor(backcolor);
            bar(46*x,35*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;

     case 'N': setcolor(backcolor);
            bar(46*x,35*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;

     default : break;
     }
}
```

```
/* PROGRAM    : q4512.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 6, 1990
   REVISED    : Apr. 6, 1990

   DESCRIPTION : This program contains the twelfth exercise about the
                 binary trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/



/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                      /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk          find_t
  #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
  #define ffblk          FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
 int in_the_exercise = 1;


/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code         */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines                    */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
  else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
}

/*********************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
      case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
      case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);      /* restore any hidden hot keys */
}



/**********************************************************************/
/* This function sets the text default values                        */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine that calls exer routine                              */
/*********************************************************************/
void main()
{
  exer();
}

/*********************************************************************/
/* Routine that asks the question, then depending on the user's answer */
/* makes necessary explanations                                      */
/*********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  12");
   /*****************************************************************/
   outtextxy(10*x,2*y,"Evaluate the following reverse Polish notation expression");
   outtextxy(30*x,3*y,"2 3 + 4 6 - - 5 * 4 +");
   /*****************************************************************/
while (in_the_exercise == 1) {
 outtextxy(15*x,14*y,"Choose one of the following, as you need :");
 outtextxy(15*x,15*y,"    a) I'm done, I want to compare my solution with yours.");
 outtextxy(15*x,16*y,"    b) I want to see step by step solution.");
 outtextxy(15*x,17*y,"    c) This is enough for me, I want to exit.");
 outtextxy(15*x,18*y,"Enter your choice here --->");
 Ch = getch ();
 if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,18*y,"    Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,35*y/2,88*x,20*y);
    setcolor(forecolor);
    }
   }
    switch (Ch)         {
   case 'a': outtextxy(47*x,18*y,"a");
     outtextxy(52*x,18*y,"You want to compare your solu-");
     outtextxy(52*x,19*y,"tion with ours. So press any  ");
     outtextxy(52*x,20*y,"key to see it.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,35*y/2,179*x/2,22*y);
     bar(2*x,4*y,179*x/2,49*y/2);
     setcolor(forecolor);
     compare_solutions();
     break;
   case 'b': outtextxy(47*x,18*y,"b");
     outtextxy(52*x,18*y,"You want to see step by step");
     outtextxy(52*x,19*y,"solution. So press any key to ");
     outtextxy(52*x,20*y,"continue.");
     Pause(30*x,24*y);
     setcolor(backcolor);
```

```
        bar(50*x,35*y/2,179*x/2,22*y);
        bar(2*x,4*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'c': outtextxy(47*x,18*y,"c");
        confirm_exit();
        break;
      default  : break;
     }
  }
  closegraph();
}




/************************************************************************/
/* This routine gives the solution to the exercise to be compared.          */
/************************************************************************/
static void compare_solutions(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
  outtextxy(35*x,10*y,"The answer is  39");
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field  again */
  bar(2*x,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/**************************************************************************/
/* This routine gives the step by step solution to the exercise          */
/**************************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(3*x/2,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
  /**********************************************************************/
  outtextxy(30*x,8*y,"  2 3 + 4 6 - - 5 * 4 +");
  Pause(30*x,24*y);
  /**********************************************************************/
  outtextxy(30*x,9*y,"= 5 4 6 - - 5 * 4 +");
  Pause(30*x,24*y);
  /**********************************************************************/
  outtextxy(30*x,10*y,"= 5 (-2) - 5 * 4 +");
  Pause(30*x,24*y);
  /**********************************************************************/
  outtextxy(30*x,11*y,"= 7 5 * 4 +");
  Pause(30*x,24*y);
  /**********************************************************************/
  outtextxy(30*x,12*y,"= 35 4 +");
  Pause(30*x,24*y);
  /**********************************************************************/
  outtextxy(30*x,13*y,"= 39");
  Pause(30*x,24*y);
  /**********************************************************************/
  setcolor(backcolor);          /* Clean the game field  again */
  bar(2*x,4*y,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,22*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,21*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
    switch (ch)         {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;

    case 'n': setcolor(backcolor);
           bar(46*x,35*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    case 'N': setcolor(backcolor);
           bar(46*x,35*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    default : break;
    }
}
```

```
/* PROGRAM   : sort.c
   AUTHOR     : Atilla BAKAN
   DATE       : Mar. 12, 1990
   REVISED    : Apr. 8, 1990

   DESCRIPTION : This program contains the tutorial for sorting and searching
                 in binary trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)           /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
```

```
  #define ff_attrib        attribute
#endif

#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions           */
static void add_shadow    (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit    (int errnum);
static void initialize    (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void Pageup        (void);
static void Pagedown      (void);
static void press_a_key   (int wrow);
static void pre_help      (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);

/* tutorial functions      */
static void sort_search       (void);
static void definition_4_6_1  (void);
static void ex_sort_1         (void);
static void ex_sort_2         (void);
static void ex_sort_3         (void);
static void ex_sort_4         (void);
static void construct         (void);
static void sorting           (void);
static void searching         (void);
static void exercises         (void);
static void exer1             (void);
static void exer2          (void);
static void exer3          (void);
static void exer4          (void);
static void exer5          (void);
static void P1            (void);
static void P2            (void);
static void P3            (void);
static void P4            (void);
static void P5            (void);
static void P6            (void);
static void P7            (void);
static void P8            (void);
static void P9            (void);
static void P10           (void);
static void P11           (void);
static void P12           (void);
static void P13           (void);
static void P14           (void);
static void P15           (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];

/*************************************************************************/
/* error message table                                                   */
/*************************************************************************/
static char *error_text[]= {
  NULL,  /* errnum =  0, no error    */
  NULL,  /* errnum == 1, windowing error */
  "Syntax:  CXLDEMO [-switches]\n\n"
     "\t -c = CGA snow elimination\n"
     "\t -b = BIOS screen writing\n"
```

```
    "\t -m = force monochrome text attributes",
   "Memory allocation error"
};

/************************************************************************/
/* miscellaneous defines                                              */
/************************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33

/************************************************************************/
/* this function will add a shadow to the active window                */
/************************************************************************/
static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}

/************************************************************************/
/* this function pops open a window and confirms that the user really   */
/* wants to quit the demo.  If so, it terminates the demo program.      */
/************************************************************************/
static void confirm_quit(void)
{
   struct _onkey_t *kblist;

   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
   add_shadow();
   wputs("\n Quit demo, are you sure? \033A\156Y\b");
   clearkeys();
   showcur();
   if(wgetchf("YN",'Y')=='Y') normal_exit();
   wclose();
   hidecur();
```

```
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}

/*****************************************************************************/
/* this function is called by the pull-down demo for a prompt                */
/*****************************************************************************/
static void disp_sure_msg(void)
{
  wprints(0,2,WHITE|_BLUE,"Are you sure?");
}

/*****************************************************************************/
/* this function handles abnormal termination. If it is passed an            */
/* error code of 1, then it is a windowing system error. Otherwise           */
/* the error message is looked up in the error message table.                */
/*****************************************************************************/
static void error_exit(int errnum)
{
  if(errnum) {
    printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
    exit(errnum);
  }
}

/*****************************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems  */
/*****************************************************************************/
static void initialize(void)
{
  /* initialize the CXL video system and save current screen info */
  videoinit();
  readcur(&crow,&ccol);
  if((savescrn=ssave())==NULL) error_exit(3);

  /* if mouse exists, turn on full mouse support */
```

```
    if(msinit()) {
        mssupport(MS_FULL);
        msgotoxy(12,49);
    }
    /* attach [Alt-X] to the confirm_quit() function */
    setonkey(0x2d00,confirm_quit,0);

    /* attach [Ctrl Pageup] to the Pageup() function */
    setonkey(0x8400,Pageup,0);

    /* attach [Ctrl Pagedown] to the Pagedown() function */
    setonkey(0x7600,Pagedown,0);

    /* initialize help system, help key = [F1] */
    whelpdef("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
              WHITE|_RED,RED|_LGREY,pre_help);
}

/*************************************************************************/
/* this function is called anytime to switch back to previous window.    */
/*************************************************************************/
static void Pageup(void)
{
    static WINDOW handle;

    handle = whandle();
    wactiv(handle - 1);
}

/*************************************************************************/
/* this function is called anytime to switch back to next window.        */
/*************************************************************************/
static void Pagedown(void)
{
    static WINDOW handle;
```

```
  handle = whandle();
  wactiv(handle + 1);
}

/**************************************************************************/
static void pre_help(void)
{
  add_shadow();
  setonkey(0x2d00,confirm_quit,0);
}

/**************************************************************************/
/* this function handles normal termination.  The original screen and cursor        */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/**************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}

/**************************************************************************/
/* this function displays a pause message then pauses for a keypress                */
/**************************************************************************/
static void press_a_key(int wrow)
{
  register int attr1;
  register int attr2;

  attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
  attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
  wcenters(wrow,attr1,"Press a key");
  wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
  hidecur();
  if(waitkey()==ESC) confirm_quit();
  wcenters(wrow,attr1,"          ");
  wprints(wrow,0,attr2,"        ");
}



/************************************************************************/
/* This routine causes short dealys during execution                   */
/************************************************************************/
static void short_delay(void)
{
  delay_(SHORT_DELAY);
}



/************************************************************************/
/* this function is called by the pull-down menu demo anytime           */
/* the  selection bar moves on or off the [Q]uit menu items.           */
/************************************************************************/
static void quit_window(void)
{
  static WINDOW handle=0;

  if(handle) {
    wactiv(handle);
    wclose();
    handle=0;
  }
  else {
    handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
    wputs(" Quit takes you back to the\n demo program's main menu.");
  }
}
```

```
/*******************************************************************/
/* shows the cursor again if it has been hidden                    */
/*******************************************************************/
static void restore_cursor(void)
{
  wtextattr(WHITE|_MAGENTA);
  showcur();
}

/*******************************************************************/
/* enlarges or shrinks the windows                                 */
/*******************************************************************/
static void size_window(int nerow,int necol)
{
  wsize(nerow,necol);
  short_delay();
}

/*******************************************************************/
/* moves the active window to a given screen coordinates           */
/*******************************************************************/
static void move_window(int nsrow,int nscol)
{
  if(wmove(nsrow,nscol)) error_exit(1);
  short_delay();
}

/*******************************************************************/
/* this routine  calls sort_search() routine whenever Pageup or Pagedown   */
/* keys are pressed.                                               */
/*******************************************************************/
void P1()
{
  wcloseall();
  sort_search();
}
```

```
/**********************************************************************/
/* this routine  calls definition 4-6-1 routine whenever Pageup or   */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P2()
{
  wcloseall();
  definition_4_6_1();
}
/**********************************************************************/
/* this routine  calls ex_sort_1 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P3()
{
  wcloseall();
  ex_sort_1();
}
/**********************************************************************/
/* this routine  calls construct routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P4()
{
  wcloseall();
  construct();
}
/**********************************************************************/
/* this routine  calls ex_sort_2 routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P5()
{
  wcloseall();
  ex_sort_2();
}
```

```
/*********************************************************************/
/* this routine  calls sorting routine whenever Pageup or           */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P6()
{
  wcloseall();
  sorting();
}
/*********************************************************************/
/* this routine  calls ex_sort_3 routine whenever Pageup or         */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P7()
{
  wcloseall();
  ex_sort_3();
}
/*********************************************************************/
/* this routine that calls searching  routine whenever Pageup or    */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P8()
{
  wcloseall();
  searching();
}
/*********************************************************************/
/* this routine that calls ex_sort_4 routine whenever Pageup or     */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P9()
{
  wcloseall();
  ex_sort_4();
}
```

```
/**********************************************************************/
/* this routine  calls exercises routine whenever Pageup or          */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P10()
{
  wcloseall();
  exercises();
}
/**********************************************************************/
/* this routine  calls exer1 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P11()
{
  wcloseall();
  exer1();
}
/**********************************************************************/
/* this routine  calls exer2 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P12()
{
  wcloseall();
  exer2();
}
/**********************************************************************/
/* this routine  calls exer3 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                         */
/**********************************************************************/
void P13()
{
  wcloseall();
  exer3();
}
```

```
/*********************************************************************/
/* this routine calls exer4 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P14()
{
  wcloseall();
  exer4();
}
/*********************************************************************/
/* this routine  calls exer5 routine whenever Pageup or             */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P15()
{
  wcloseall();
  exer5();
}

/*********************************************************************/
/* main routine, calls minimal spanning tree tutorial                */
/*********************************************************************/
void main()
{
 initialize();
 sort_search();
}
```

```
/************************************************************************/
/* This routine  calls definition, example and algorithm routines about       */
/* sorting and searching in binary trees.                                     */
/************************************************************************/
static void sort_search(void)
{
  register int *scrn;

  if((scrn=ssave())==NULL) error_exit(3);
  cclrscrn(LGREY|_BLUE);
  /**********************************************************************/
  /* attach [Pagedown] to the definition_4_6_1() function */
  setonkey(0x5100,P2,0);
  /**********************************************************************/
  if((w[1]=wopen(5,20,15,60,3,WHITE|_BLACK,RED|_CYAN))==0) error_exit(1);
  wtitle("[Sorting and Searching]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  Maintaining a large data set is a common problem for data"
         " processors. This consists not only of updating the data set"
         " by adding and deleting, but also of searching the data for"
         " a particular for a particular piece of of information.");
  press_a_key(8);
  wslide(0,0);
  short_delay();
  /**********************************************************************/
  if((w[2]=wopen(5,20,18,60,3,WHITE|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Binary Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  Suppose, for instance, that a company X maintains a list"
          " of its customers. When an order is received, the company must"
          " search this list to determine if the order is from an old or"
          " a new customer. If the order is from a new customer, then"
          " this customer's name must be added to the list. Moreover,"
          " when a customer goes out of business, that customer's name"
          " must be removed from the list.");
press_a_key(11);
wslide(11,0);
short_delay();
/*********************************************************************/
if((w[3]=wopen(5,20,17,60,3,WHITE|_BLACK,RED|_CYAN))==0) error_exit(1);
wtitle("[Sorting and Searching]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw(" One way, maybe the easiest one, to maintain such a list is"
          " to keep the data, let's say in an array in the order in which"
          " they are receiwed. As we said above, this method enables items"
          " to be added to the list easily; for instance, if a new name"
          " to be added, this name can be added to the end of the existing"
          " array.");
press_a_key(10);
wslide(0,39);
short_delay();
/*********************************************************************/
if((w[4]=wopen(5,20,17,60,3,WHITE|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
wtitle("[Sorting and Searching]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw("  However, this method makes it very time consuming"
          " to determine if a particular name is in the list. In this"
          " case, to be able to find this  name , the whole array"
          " is to be gone through. If the size of the array is small"
          " there is no problem, but, if we are talking about a big"
          " company, then we are in trouble.");
press_a_key(10);
wslide(12,39);
short_delay();
/*********************************************************************/
```

```
if((w[5]=wopen(5,15,16,65,3,WHITE|_BLACK,WHITE|_BLUE))==0)
          error_exit(1);
wtitle("[Sorting and Searching]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw("  Another approach is to keep the list in alphabetical order."
          " So it would be easier to search the list for a particular"
          " name. However, adding or deleting from the list is more"
          " difficult because of the need to reposition the entries when"
          " an item is added or deleted. Like previous one, this process"
          " is prohibitive if the list is very long.");
press_a_key(9);
wslide(0,20);
short_delay();
/*******************************************************************/
if((w[6]=wopen(5,15,17,65,3,WHITE|_BLACK,WHITE|_BLUE))==0)
          error_exit(1);
wtitle("[Sorting and Searching]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw("  The other approach is to store the data at the vertices of"
          " a binary tree.");
wputs("\n           How is this done ?\n");
wputsw("  Actually, we came to the topic of this section which we"
          " intentionally left to the last.");
wputs("\n\n    Sorting and searching in binary trees.\n\n");
wputsw("  To do this we've got to talk about a new concept,"
          "   'binary search trees'.");
press_a_key(10);
wslide(12,20);
/*******************************************************************/
short_delay();
wcloseall();
definition_4_6_1();
srestore(scrn);
}
```

```
/************************************************************************/
/* This routine gives the definition of a binary search tree             */
/************************************************************************/
static void definition_4_6_1(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the sort_search() function */
  setonkey(0x4900,P1,0);
  /*********************************************************************/
  /* attach [Pagedown] to the ex_sort_1() function */
  setonkey(0x5100,P3,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,19,65,3,WHITE|_BLACK,WHITE|_GREEN))==0)
          error_exit(1);
  wtitle("[Binary Search Trees - Definition 4_6_1]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  A binary search tree for the list is a binary tree in which"
          " each vertex is labeled by an element of the list such that:");
  wputs("\n  (1) No two vertices have the same label\n");
  wputsw("  (2) If vertex U belongs to the left subtree of vertex V,"
          " then U <= V.");
  wputs("\n");
  wputsw("  (3) If vertex W belongs to the right subtree of vertex V,"
          " then V <= W.");
  wputs("\n");
  wputsw("  Thus , for each vertex V, all descendants of V in the left"
          " subtree of V precede V, and all descendants of V in the right"
          " subtree of V follow V.");
  press_a_key(12);
  short_delay();
  wslide(0,0);
  short_delay();
  ex_sort_1();
}
```

```
/*********************************************************************/
/* This routine gives an example for typical binary search trees          */
/*********************************************************************/
static void ex_sort_1 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the definition_4_6_1() function */
  setonkey(0x4900,P2,0);
  /*******************************************************************/
  /* attach [Pagedown] to the construct() function */
  setonkey(0x5100,P4,0);
  /*******************************************************************/
  if((w[2]=wopen(5,15,10,65,3,WHITE|_BLACK,WHITE|_LGREY))==0)
          error_exit(1);
  wtitle("[Binary Search Trees - Example_4_1]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw("  Do you want to see an example ?");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"examp461.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  construct();
}
```

```
/*****************************************************************/
/* This routine gives the algorithm to construct  a binary search tree              */
/*****************************************************************/
static void construct(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the ex_sort_1() function */
  setonkey(0x4900,P3,0);
  /*****************************************************************/
  /* attach [Pagedown] to the ex_sort_2() function */
  setonkey(0x5100,P5,0);
  /*****************************************************************/
  if((w[1]=wopen(5,15,13,65,3,WHITE|_BLACK,WHITE|_RED))==0)
            error_exit(1);
  wtitle("[Binary Search Tree Construction]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  There is a systematic way to construct a binary search tree"
          " for a list. The basic idea is to put smaller elements as left"
          " children and larger elements as right children.");
  wputs("\n");
  wputsw("  Following is the algorithm to construct a binary search tree"
          " from a given list.");
  press_a_key(6);
  wclose();
  short_delay();
  /*****************************************************************/
  if((w[2]=wopen(0,10,24,65,3,WHITE|_BLACK,BLACK|_CYAN))==0)
            error_exit(1);
  wtitle("[Binary Search Tree Construction Algorithm]",
          TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  This algorithm constructs a binary search tree with"
          " vertices labeled A1, A2,.....,An, where A1, A2,...,An are"
          " distinct. In the algorithm we refer to a vertex by its label.");
  wputs("\n");
  wputsw("  Step 1 (start). Construct a root and label it A1. If n = 1,"
            " we are done; otherwise, let V = A1 and k = 2.");
  wputs("\n");
  wputsw("  Step 2 (if smaller, go left). If V <= A1, go to Step 3."
            " Otherwise, we have Ak <= V.");
  wputs("\n");
  wputsw("    (a) If V has no left child, construct a left child L for"
            " V and label L with Ak. If k = n, we are done; otherwise,"
            " increase k by 1, set V = A1, and go to Step 2.");
  wputs("\n");
  wputsw("    (b) Otherwise, if V has a left child L, set V = L, and"
            " go to Step 2.");
  wputs("\n");
  wputs("  Step 3 (if larger, go right). We have V <= Ak .\n");
  wputsw("    (a) If V has no right child, construct a right child R for"
            " V and label R with Ak. If k = n, we are done; otherwise,"
            " increase k by 1, set V = A1, and go to Step 2.");
  wputs("\n");
  wputsw("    (b) Otherwise, if V has a right child R, set V = R, and"
            " go to Step 2.");
  press_a_key(22);
  short_delay();
  ex_sort_2();
  short_delay();
}
```

```
/*************************************************************************/
/* This routine gives an example of a binary search tree  construction         */
/*************************************************************************/
static void ex_sort_2 (void)
{
  /*********************************************************************/
  /* attach [Pageup] to the construct() function */
  setonkey(0x4900,P4,0);
  /*********************************************************************/
  /* attach [Pagedown] to the sorting() function */
  setonkey(0x5100,P6,0);
  /*********************************************************************/
  if((w[3]=wopen(5,15,10,65,3,WHITE|_BLACK,WHITE|_LGREY))==0)
              error_exit(1);
  wtitle("[Binary Search Trees - Example_4_2]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw("  We need to show an example.");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"examp462.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  sorting();
}
```

```
/***************************************************************************/
/* This routine teaches how a given list is sorted by using binary search  trees.      */
/***************************************************************************/
static void sorting(void)
{
  /************************************************************************/
  /* attach [Pageup] to the ex_sort_2() function */
  setonkey(0x4900,P5,0);
  /************************************************************************/
  /* attach [Pagedown] to the ex_sort_3() function */
  setonkey(0x5100,P7,0);
  /************************************************************************/
  if((w[1]=wopen(5,15,17,54,3,WHITE|_BLACK,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Sorting with Binary Search Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  While we were introducing this section we talked about"
         " sorting. As you all know, there are so many sorting"
         " techniques. Since our particular concern is on binary"
        " search trees for now, we will show you how binary trees can"
        " be used for having sorted list of elements and won't cover"
        " any other sorting technique.");
  press_a_key(10);
  wslide(0,0);
  short_delay();
  /************************************************************************/
  if((w[2]=wopen(0,15,24,54,3,WHITE|_BLACK,RED|_LGREY))==0)
          error_exit(1);
  wtitle("[Sorting with Binary Search Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  It wouldn't be a good idea to use binary search trees"
         " just for sorting purposes, since there are better sorting"
         " techniques.For instance, if large databases are in question,"
         " it might be costly to use binary trees. Because, to build the"
```

```
            " tree then to make a traversal on each one of the vertexes (since"
            " we are talking about having a sorted print out of the list, we"
            " have to look at each distinct element,an print it somehow.)"
            " will require us to visit the same vertex at least twice."
            " But on the other hand, if have already been maintaining the"
            " records in a binary search tree and we are asked to give a"
            " sorted list of, let's say employees, by their last names"
           " this technique which we are about to talk about, will no"
           " doubt be helpful, moreover will be necessary.");
    press_a_key(22);
    wslide(0,39);
    short_delay();
    /*****************************************************************/
    if((w[3]=wopen(5,15,14,54,3,WHITE|_BLACK,RED|_GREEN))==0)
              error_exit(1);
    wtitle("[Sorting with Binary Search Trees]",TCENTER,_LGREY|BROWN);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputsw(" This so-called sorting technique with binary search trees"
             " is no more than making an inorder traversal in a binary search"
             " tree. As you all remember, in this traversal we were visiting"
             " the vertices in the left_child-parent-right_child order.");
    press_a_key(7);
    wslide(13,0);
    short_delay();
    ex_sort_3();
  }
```

```
/**********************************************************************/
/* This routine gives an inorder traversal of a binary search tree  and          */
/* consequently prints the sorted list of the elements in the tree.              */
/**********************************************************************/
static void ex_sort_3 (void)
{
  /*******************************************************************/
  /* attach [Pageup] to the sorting() function */
  setonkey(0x4900,P6,0);
  /*******************************************************************/
  /* attach [Pagedown] to the searching() function */
  setonkey(0x5100,P8,0);
  /*******************************************************************/
  if((w[4]=wopen(5,15,10,65,3,WHITE|_BLACK,RED|_BLACK))==0)
          error_exit(1);
  wtitle("[Sorting with Binary Search Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" We feel like its a good time for an example. Do You ?");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"examp463.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  searching();
}
```

```
/*************************************************************************/
/* This routine teaches how a particular element is searched in a binary          */
/*  search tree.                                                                   */
/*************************************************************************/
static void searching(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the ex_sort_3() function */
  setonkey(0x4900,P7,0);
  /*************************************************************************/
  /* attach [Pagedown] to the ex_sort_4() function */
  setonkey(0x5100,P9,0);
  /*************************************************************************/
  if((w[1]=wopen(5,15,12,65,3,WHITE|_BLACK,WHITE|_CYAN))==0)
          error_exit(1);
  wtitle("[Sorting with Binary Search Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  We now came to the last topic of this section, 'searching'"
          " The search algorithm is quite simple. That's why we give"
          " a verbal description and leave it to you to express search"
          " as a formal recursive algorithm.");
  press_a_key(5);
  wslide(0,15);
  /*************************************************************************/
  if((w[2]=wopen(5,15,15,65,3,WHITE|_BLACK,BLACK|_GREEN))==0)
          error_exit(1);
  wtitle("[Sorting with Binary Search Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("  The algorithm works like this :\n");
  wputsw("  Start with the root of the tree, if the search key equals"
          " the vertex key, the search halts.");
  wputs("\n");
  wputsw(" If the search key is less than the vertex key, the left subtree"
          " is searched, if it is not empty.");
  wputsw(" Otherwise, the right subtree is searched, if it is not empty.");
  press_a_key(8);
  wslide(8,15);
  short_delay();
  ex_sort_4();
}


/**************************************************************************/
/* This routine gives an example of search implementation on a binary    */
/* search tree.                                                           */
/**************************************************************************/
static void ex_sort_4 (void)
{
  /************************************************************************/
  /* attach [Pageup] to the searching() function        */
  setonkey(0x4900,P8,0);
  /************************************************************************/
  /* attach [Pagedown] to the exercises() function */
  setonkey(0x5100,P10,0);
  /************************************************************************/
  if((w[3]=wopen(5,10,10,70,3,WHITE|_BLACK,RED|_BLACK))==0)
         error_exit(1);
  title("[Sorting with Binary Search Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n You see, it is very easy. Now how about an example ?");
  press_a_key(3);
  wslide(19,10);
  short_delay();
  /************************************************************************/
  wcloseall();
  spawnl(P_WAIT,"examp464.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exercises();
}
```

```
/*******************************************************************/
/* This routine makes a small quiz about the binary search trees.      */
/*******************************************************************/
void exercises(void)
 {
   register int *screen;

   /*******************************************************************/
   /* attach [Pageup] to the ex_sort_4() function             */
   setonkey(0x4900,P9,0);
   /*******************************************************************/
   /* attach [Page  wn] to the exer1() function */
   setonkey(0x5100,P11,0);
   /*******************************************************************/
   if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
              error_exit(1);
   wtitle("[Binary Search Trees]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   add_shadow();
   wputs("\n");
   wputsw(" We have completed our presentation of this section. Are"
             " you ready for a pop quiz ? ");
   press_a_key(3);
   short_delay();
   wclose();
   if((screen=ssave())==NULL) error_exit(3); {
   /*******************************************************************/
    exer1();
   /* if mouse exists, turn on full mouse support */
     if(msinit()) {
      mssupport(MS_FULL);
      msgotoxy(12,49);
      }
   }
   srestore(screen);
 }
```

```
/*******************************************************************/
/* Dummy function to call the actual exercise 4.6.1                  */
/*******************************************************************/
static void exer1(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the ex_sort_4() function          */
  setonkey(0x4900,P9,0);
  /*****************************************************************/
  /* attach [Pagedown] to the exer2() function */
  setonkey(0x5100,P12,0);
  /*****************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Search Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the first question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q461.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer2();
}
```

```
/************************************************************************/
/* Dummy function to call the actual exercise 4.6.2                   */
/************************************************************************/
static void exer2(void)
{
  /**********************************************************************/
  /* attach [Pageup] to the exer1() function        */
  setonkey(0x4900,P11,0);
  /**********************************************************************/
  /* attach [Pagedown] to the exer3() function */
  setonkey(0x5100,P13,0);
  /**********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Search Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the second question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q462.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer3();
}
```

```
/***********************************************************************/
/* Dummy function to call the actual exercise 4.6.3                    */
/***********************************************************************/
static void exer3(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the exer2() function          */
  setonkey(0x4900,P12,0);
  /*********************************************************************/
  /* attach [Pagedown] to the exer4() function */
  setonkey(0x5100,P14,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Search Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("       Here is the third question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q463.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer4();
}
```

```
/**********************************************************************/
/* Dummy function to call the actual exercise 4.6.4                   */
/**********************************************************************/
static void exer4(void)
{
  /********************************************************************/
  /* attach [Pageup] to the exer3() function           */
  setonkey(0x4900,P13,0);
  /********************************************************************/
  /* attach [Pagedown] to the exer5() function */
  setonkey(0x5100,P15,0);
  /********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
        error_exit(1);
  wtitle("[Binary Search Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the forth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q464.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer5();
}
```

```
/*********************************************************************/
/* Dummy function to call the actual exercise 4.6.5                  */
/*********************************************************************/
static void exer5(void)
{
  /******************************************************************/
  /* attach [Pageup] to the exer4() function              */
  setonkey(0x4900,P14,0);
  /******************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Binary Search Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the fifth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q465.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  normal_exit();
}
```

```
/* PROGRAM   : examp461.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for a binary seach tree.



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                     /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                    /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/* Utility functions        */
static void init_graph   (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer           (void);

/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;




/*********************************************************************/
/* This function is used for including drivers to the executable code  */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*******************************************************************/
  settext();
  /*******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       }
  forecolor = WHITE;
 }
```

```
/************************************************************************/
/* This function sets the text default values                          */
/************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/************************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
  }



/************************************************************************/
/* main routine, calls exer routine                                     */
/************************************************************************/
void main()
{
  exer();
}
```

```
/**********************************************************************/
/* This routine illustrates a binary search tree                       */
/**********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-6-1");
  /********************************************************************/
  pieslice(45*x,4*y,0,359,2);   /* Mary     */
  pieslice(30*x,7*y,0,359,2);   /* Hande    */
  pieslice(60*x,7*y,0,359,2);   /* Tom      */
  moveto(30*x,7*y);  lineto(45*x,4*y);  lineto(60*x,7*y);
  outtextxy(42*x,7*y/2,"Mary");
  outtextxy(23*x,7*y,"Hande");
  outtextxy(61*x,7*y,"Tom");
  pieslice(20*x,10*y,0,359,2);  /* Atilla   */
  pieslice(40*x,10*y,0,359,2);  /* Kim      */
  pieslice(50*x,10*y,0,359,2);  /* Pat      */
  pieslice(70*x,10*y,0,359,2);  /* Yavuz    */
  outtextxy(16*x,21*y/2,"Atilla");
  outtextxy(35*x,10*y,"Kim");
  outtextxy(52*x,10*y,"Pat");
  outtextxy(67*x,21*y/2,"Yavuz");
  moveto(20*x,10*y);  lineto(30*x,7*y); lineto(40*x,10*y);
  moveto(50*x,10*y);  lineto(60*x,7*y); lineto(70*x,10*y);
  pieslice(35*x,13*y,0,359,2);  /* Hasene  */
  pieslice(45*x,13*y,0,359,2);  /* Mantak  */
  pieslice(55*x,13*y,0,359,2);  /* Sam      */
  outtextxy(31*x,27*y/2,"Hasene");
  outtextxy(42*x,27*y/2,"Mantak");
  outtextxy(54*x,27*y/2,"Sam");
  moveto(35*x,13*y);  lineto(40*x,10*y); lineto(45*x,13*y);
  moveto(50*x,10*y);  lineto(55*x,13*y);
```

```
outtextxy(2*x,15*y,"In this example of binary search tree every intermediate ver-
                    tex is alphabet-");
outtextxy(2*x,16*y,"ically greater than its left child and less than its right child");
outtextxy(2*x,17*y,"Adding a new name to tree is simple because we need only in-
                    clude one vertex");
outtextxy(2*x,18*y,"and edge in the tree, and also searching this tree for a particu-
                    lar name");
outtextxy(2*x,19*y,"requires no more than four comparisons if we search the tree
                    properly");
/******************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(0,0,MaxX,MaxY);
setcolor(forecolor);
rectangle(x,y/2,MaxX-x,MaxY-y/2);
/******************************************************************/
outtextxy(2*x,2*y,"In this example you  see two possible binary search trees for ");
outtextxy(2*x,3*y,"            1, 2, 3, 4, 5, 6, 8, 9, 10");
Pause(30*x,24*y);
/******************************************************************/
pieslice(25*x,6*y,0,359,2);   /* 6 */
pieslice(15*x,9*y,0,359,2);   /* 4 */
pieslice(35*x,9*y,0,359,2);   /* 9 */
moveto(15*x,9*y);  lineto(25*x,6*y);  lineto(35*x,9*y);
outtextxy(25*x,11*y/2,"6");
outtextxy(13*x,9*y,"4");
outtextxy(36*x,9*y,"9");
pieslice(5*x,12*y,0,359,2);  /* 1  */
pieslice(20*x,12*y,0,359,2);  /* 5  */
pieslice(30*x,12*y,0,359,2);  /* 8  */
pieslice(45*x,12*y,0,359,2);  /* 10 */
outtextxy(3*x,12*y,"1");
outtextxy(20*x,25*y/2,"5");
outtextxy(30*x,25*y/2,"8");
outtextxy(45*x,25*y/2,"10");
moveto(5*x,12*y);  lineto(15*x,9*y); lineto(20*x,12*y);
```

```
  moveto(30*x,12*y); lineto(35*x,9*y); lineto(45*x,12*y);
  pieslice(10*x,15*y,0,359,2);  /* 2 */
  outtextxy(11*x,31*y/2,"2");
  moveto(5*x,12*y); lineto(10*x,15*y);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  pieslice(75*x,6*y,0,359,2);   /* 8 */
  pieslice(70*x,8*y,0,359,2);   /* 6 */
  pieslice(80*x,8*y,0,359,2);   /* 9 */
  moveto(70*x,8*y); lineto(75*x,6*y); lineto(80*x,8*y);
  outtextxy(75*x,11*y/2,"8");
  outtextxy(68*x,8*y,"6");
  outtextxy(82*x,8*y,"9");
  pieslice(67*x,10*y,0,359,2);   /* 5 */
  pieslice(64*x,12*y,0,359,2);   /* 4 */
  pieslice(61*x,14*y,0,359,2);   /* 2 */
  pieslice(58*x,16*y,0,359,2);   /* 1 */
  pieslice(83*x,10*y,0,359,2);   /* 10 */
  moveto(70*x,8*y); lineto(67*x,10*y); lineto(64*x,12*y);
  lineto(61*x,14*y); lineto(58*x,16*y);
  moveto(80*x,8*y); lineto(83*x,10*y);
  outtextxy(65*x,10*y,"5");
  outtextxy(62*x,12*y,"4");
  outtextxy(59*x,14*y,"2");
  outtextxy(58*x,33*y/2,"1");
  outtextxy(83*x,21*y/2,"10");
  /*******************************************************************/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
}
```

```
/* PROGRAM   : examp462.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for constructing a binary
                 search tree.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                  /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                    /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk          find_t
  #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name          name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);

/**********************************************************************/
/* graphic initialization variables                                   */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/**********************************************************************/
/* This function is used for including drivers to the executable code  */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines            */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE,
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/**************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
```

```
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}
/***********************************************************************/
/* This function sets the text default values                           */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen               */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/***********************************************************************/
/* main routine, calls exer routine                                     */
/***********************************************************************/
void main()
{
  exer();
}
```

```
/***********************************************************************/
/* This routine illustrates construction of a binary search tree.          */
/***********************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE 4-6-2");
   /*******************************************************************/
   outtextxy(2*x,2*y,"Now we will try to give you an example about binary search
                           tree construction");
   outtextxy(2*x,3*y,"We will show you how we applied the binary search tree con-
                           struction algorithm");
   outtextxy(2*x,4*y,"on the list  'H, F, N, D, G, L, O, B, E, J, M, A, C, I, K' step by
                            step.");
   /*******************************************************************/
   outtextxy(44*x,5*y,"Step by step appliction of the Alg.");
   moveto(43*x,11*y/2);  lineto(89*x,11*y/2);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,70*x,49*y/2);
   setcolor(forecolor);
   /*******************************************************************/
   outtextxy(44*x,6*y,"Begin construction starting with the");
   outtextxy(44*x,7*y,"first letter in the list H. Construct");
   outtextxy(44*x,8*y,"the root and label it with H.");
   pieslice(27*x,5*y,0,359,2);    /* H */
   outtextxy(27*x,9*y/2,"H");
   setlinestyle(3,0,1);
   moveto(44*x,17*y/2); lineto(89*x,17*y/2);
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,70*x,49*y/2);
```

```
setcolor(forecolor);
/***********************************************************************/
outtextxy(44*x,9*y,"Take the next element F from the list.");
outtextxy(44*x,10*y,"Compare it with root, it is smaller so");
outtextxy(44*x,11*y,"go left. There is no left child, const-");
outtextxy(44*x,12*y,"ruct it and label it with F.");
pieslice(17*x,7*y,0,359,2);   /* F */
outtextxy(15*x,7*y,"F");
moveto(17*x,7*y);  lineto(27*x,5*y);
setlinestyle(3,0,1);
moveto(44*x,25*y/2); lineto(89*x,25*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/***********************************************************************/
outtextxy(44*x,13*y,"Take the next element N from the list.");
outtextxy(44*x,14*y,"Compare it with root, it is larger so");
outtextxy(44*x,15*y,"go right. There is no right child, con-");
outtextxy(44*x,16*y,"struct it and label it with N.");
pieslice(37*x,7*y,0,359,2);   /* N */
outtextxy(39*x,7*y,"N");
moveto(27*x,5*y);  lineto(37*x,7*y);
setlinestyle(3,0,1);
moveto(44*x,33*y/2); lineto(89*x,33*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/***********************************************************************/
outtextxy(44*x,17*y,"Take D from the list. Compare it with");
outtextxy(44*x,18*y,"the root, it is smaller so go left.");
outtextxy(44*x,19*y,"There is a left child so this time ");
outtextxy(44*x,20*y,"compare D with left child F. Since");
outtextxy(44*x,21*y,"D <= F go left again. There is no left");
outtextxy(44*x,22*y,"child, construct it, then label it with");
outtextxy(44*x,23*y,"D.");
pieslice(12*x,9*y,0,359,2);   /* D */
outtextxy(10*x,9*y,"D");
moveto(12*x,9*y); lineto(17*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,6*y,"Take G from the list. Compare it with");
outtextxy(44*x,7*y,"the root; it is smaller, then go left,");
outtextxy(44*x,8*y,"it is greater than left child, go right");
outtextxy(44*x,9*y,"there is no right child, construct it");
outtextxy(44*x,10*y,"and label it with G.");
pieslice(22*x,9*y,0,359,2);   /* G */
outtextxy(22*x,19*y/2,"G");
moveto(17*x,7*y);   lineto(22*x,9*y);
setlinestyle(3,0,1);
moveto(44*x,21*y/2); lineto(89*x,21*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,11*y,"Take L from the list, compare it with");
outtextxy(44*x,12*y,"the root; it is larger, then go right,");
outtextxy(44*x,13*y,"it is smaller than right child, go left");
outtextxy(44*x,14*y,"there is no left child, construct it");
outtextxy(44*x,15*y,"and label it with L.");
pieslice(32*x,9*y,0,359,2);   /* L */
outtextxy(30*x,9*y,"L");
moveto(32*x,9*y); lineto(37*x,7*y);
```

```
setlinestyle(3,0,1),
moveto(44*x,31*y/2); lineto(89*x,31*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,16*y,"Take O from the list, compare it with");
outtextxy(44*x,17*y,"the root; it is larger, then go right,");
outtextxy(44*x,18*y,"it is larger than right child, go right");
outtextxy(44*x,19*y,"there is no right child, construct it");
outtextxy(44*x,20*y,"and label it with O.");
pieslice(42*x,9*y,0,359,2);   /* O */
outtextxy(42*x,19*y/2,"O");
moveto(37*x,7*y);  lineto(42*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,6*y,"Take B from the list, compare it with");
outtextxy(44*x,7*y,"the root(H); it is smaller, then go left,");
outtextxy(44*x,8*y,"it is smaller than left child(F),go left");
outtextxy(44*x,9*y,"it is still smaller than left child(D)");
outtextxy(44*x,10*y,"then go left again. But as you see D");
outtextxy(44*x,11*y,"has no left child, so construct one");
outtextxy(44*x,12*y,"and label it with B.");
pieslice(7*x,11*y,0,359,2);    /* B */
outtextxy(5*x,11*y,"B");
moveto(7*x,11*y);  lineto(12*x,9*y);
setlinestyle(3,0,1);
moveto(44*x,25*y/2); lineto(89*x,25*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
```

```
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,13*y,"Take E from the list, compare it with");
outtextxy(44*x,14*y,"the root(H); it is smaller, then go left");
outtextxy(44*x,15*y,"it is smaller than left child(F),go left");
outtextxy(44*x,16*y,"it is larger than left child(D),go right");
outtextxy(44*x,17*y,"But as you see D has no right child, so");
outtextxy(44*x,18*y," construct one and label it with E.");
pieslice(17*x,11*y,0,359,2);   /* E */
outtextxy(17*x,23*y/2,"E");
moveto(12*x,9*y);  lineto(17*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,6*y,"Take J from the list, compare it with");
outtextxy(44*x,7*y,"the root(H); it is larger, then go right");
outtextxy(44*x,8*y,"it is smaller than right child(N),go left");
outtextxy(44*x,9*y,"it is still smaller than left child(L)");
outtextxy(44*x,10*y,"then go left again. But as you see L");
outtextxy(44*x,11*y,"has no left child, so construct one");
outtextxy(44*x,12*y,"and label it with J.");
pieslice(27*x,11*y,0,359,2);   /* J */
outtextxy(24*x,11*y,"J");
moveto(27*x,11*y);  lineto(32*x,9*y);
setlinestyle(3,0,1);
moveto(44*x,25*y/2); lineto(89*x,25*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
```

```
/*******************************************************************/
outtextxy(44*x,13*y,"Take M from the list, compare it with");
outtextxy(44*x,14*y,"the root(H);it is larger, then go right");
outtextxy(44*x,15*y,"again,it is smaller than right child(N)");
outtextxy(44*x,16*y,"go left, but it is larger than left");
outtextxy(44*x,17*y,"child(L), go right. But as you see L,has");
outtextxy(44*x,18*y,"no right child, so construct one and");
outtextxy(44*x,19*y,"label it with M.");
pieslice(37*x,11*y,0,359,2);   /* M */
outtextxy(37*x,23*y/2,"M");
moveto(32*x,9*y); lineto(37*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"Take A from the list, compare it with");
outtextxy(44*x,7*y,"the root(H); it is smaller, then go left");
outtextxy(44*x,8*y,"it is smaller than left child(F),go left");
outtextxy(44*x,9*y,"it is still smaller than left child(D)");
outtextxy(44*x,10*y,"then go left again. Again it is smaller");
outtextxy(44*x,11*y,"than left child(B) so go left, but B");
outtextxy(44*x,12*y,"has no left child, so construct one");
outtextxy(44*x,13*y,"and label it with A.");
pieslice(2*x,13*y,0,359,2);    /* A */
outtextxy(2*x,27*y/2,"A");
moveto(2*x,13*y); lineto(7*x,11*y);
setlinestyle(3,0,1);
moveto(44*x,27*y/2); lineto(89*x,27*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
```

```
outtextxy(44*x,14*y,"Take C from the list, compare it with");
outtextxy(44*x,15*y,"the root(H); it is smaller, then go left");
outtextxy(44*x,16*y,"it is smaller than left child(F),go left");
outtextxy(44*x,17*y,"it is smaller than left child(D),go left");
outtextxy(44*x,18*y,"Again it is larger than left child(B) so");
outtextxy(44*x,19*y,"go right, but B  has no right child, so");
outtextxy(44*x,20*y,"construct one and label it with C.");
pieslice(12*x,13*y,0,359,2);    /* C */
outtextxy(12*x,27*y/2,"C");
moveto(7*x,11*y);  lineto(12*x,13*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,6*y,"Take I from the list, compare it with");
outtextxy(44*x,7*y,"the root(H); it is larger, then go right,");
outtextxy(44*x,8*y,"it is smaller than right child(N),go left");
outtextxy(44*x,9*y,"it is still smaller than left child(L)");
outtextxy(44*x,10*y,"then go left again. But as you see I");
outtextxy(44*x,11*y,"is still smaller than J, go left again.");
outtextxy(44*x,12*y,"But J has no left child, so construct ");
outtextxy(44*x,13*y,"one and label it with I.");
pieslice(22*x,13*y,0,359,2);    /* I */
outtextxy(22*x,27*y/2,"I");
moveto(22*x,13*y);  lineto(27*x,11*y);
setlinestyle(3,0,1);
moveto(44*x,27*y/2); lineto(89*x,27*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,14*y,"Take K from the list, compare it with");
outtextxy(44*x,15*y,"the root(H); it is larger, then go right");
outtextxy(44*x,16*y,"again, it is smaller than right child(N)");
outtextxy(44*x,17*y,"go left, it is still smaller than left");
outtextxy(44*x,18*y,"child(L), go right. This time it is lar-");
outtextxy(44*x,19*y,"ger than left child(J), so go right.But");
outtextxy(44*x,20*y,"J has no right child, so construct one");
outtextxy(44*x,21*y,"and label it with K.");
pieslice(32*x,13*y,0,359,2);   /* K */
outtextxy(32*x,27*y/2,"K");
moveto(27*x,11*y); lineto(32*x,13*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(44*x,6*y,"As you see, we constructed a binary tree");
outtextxy(44*x,7*y,"with the given list of letters. Further-");
outtextxy(44*x,8*y,"more, labels for the left descendants");
outtextxy(44*x,9*y,"(those on the left side) are smaller than");
outtextxy(44*x,10*y,"the label for the parent, and labels for");
outtextxy(44*x,11*y,"the right descendants are larger. Thus");
outtextxy(44*x,12*y,"by using the algorithm, we did construc-");
outtextxy(44*x,13*y,"a binary search tree.");
/*****************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM    : examp463.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 18, 1990
   REVISED    : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for  sorting a given list us-
                 ing bnary search tree.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                   /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/* Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int i);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions    */
static void exer            (void);

/************************************************************************/
/* graphic initialization variables                                      */
/************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/************************************************************************/
/* This function is used for including drivers to the executable code    */
/************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
    case 'y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
```

```
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/**********************************************************************/
/* This function sets the text default values                          */
/**********************************************************************/
static void settext(void)
{
  settcxtstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/**********************************************************************/
/* main routine, calls exer routine                                    */
/**********************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates sorting via a binary search tree.        */
/*********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-6-3");
  /*******************************************************************/
  outtextxy(2*x,2*y,"If you remember, we have used this example both while we
                    were talking about");
  outtextxy(2*x,3*y,"traversals and constructing binary search trees. We insist on
                    using the same");
  outtextxy(2*x,4*y,"example, because we think that it will be helpful for you to com-
                    pare");
  /*******************************************************************/
  pieslice(27*x,5*y,0,359,2);   /* H */
  pieslice(17*x,7*y,0,359,2);   /* F */
  pieslice(37*x,7*y,0,359,2);   /* N */
  moveto(17*x,7*y);  lineto(27*x,5*y);  lineto(37*x,7*y);
  outtextxy(27*x,9*y/2,"H");
  outtextxy(15*x,7*y,"F");
  outtextxy(39*x,7*y,"N");
  pieslice(12*x,9*y,0,359,2);   /* D */
  pieslice(22*x,9*y,0,359,2);   /* G */
  pieslice(32*x,9*y,0,359,2);   /* L */
  pieslice(42*x,9*y,0,359,2);   /* O */
  moveto(12*x,9*y);  lineto(17*x,7*y);  lineto(22*x,9*y);
  moveto(32*x,9*y);  lineto(37*x,7*y);  lineto(42*x,9*y);
  outtextxy(10*x,9*y,"D");
  outtextxy(22*x,19*y/2,"G");
  outtextxy(30*x,9*y,"L");
  outtextxy(42*x,19*y/2,"O");
  pieslice(7*x,11*y,0,359,2);   /* B */
pieslice(17*x,11*y,0,359,2);   /* E */
pieslice(27*x,11*y,0,359,2);   /* J */
pieslice(37*x,11*y,0,359,2);   /* M */
moveto(7*x,11*y);  lineto(12*x,9*y);  lineto(17*x,11*y);
moveto(27*x,11*y); lineto(32*x,9*y);  lineto(37*x,11*y);
outtextxy(5*x,11*y,"B");
outtextxy(17*x,23*y/2,"E");
outtextxy(24*x,11*y,"J");
outtextxy(37*x,23*y/2,"M");
pieslice(2*x,13*y,0,359,2);    /* A */
pieslice(12*x,13*y,0,359,2);   /* C */
pieslice(22*x,13*y,0,359,2);   /* I */
pieslice(32*x,13*y,0,359,2);   /* K */
moveto(2*x,13*y);  lineto(7*x,11*y);  lineto(12*x,13*y);
moveto(22*x,13*y); lineto(27*x,11*y);  lineto(32*x,13*y);
outtextxy(2*x,27*y/2,"A");
outtextxy(12*x,27*y/2,"C");
outtextxy(22*x,27*y/2,"I");
outtextxy(32*x,27*y/2,"K");
/*******************************************************************/
outtextxy(44*x,5*y,"Notes");
moveto(43*x,11*y/2); lineto(89*x,11*y/2);
outtextxy(3*x,29*y/2,"Inorder  Listing");
moveto(2*x,15*y);  lineto(40*x,15*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
outtextxy(44*x,6*y,"Since you all know what we are doing,");
outtextxy(44*x,7*y,"this time we won't tell you the detailed");
outtextxy(44*x,8*y,"steps. We want to draw your attention");
outtextxy(44*x,9*y,"on the outcoming inorder listing.");
outtextxy(3*x,16*y,"A");
Pause(30*x,24*y);
setcolor(backcolor);
```

```
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(5*x,16*y,"B");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(7*x,16*y,"C");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(9*x,16*y,"D");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(11*x,16*y,"E");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(13*x,16*y,"F");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(15*x,16*y,"G");
Pause(30*x,24*y);
setcolor(backcolor);
```

```
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(17*x,16*y,"H");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(19*x,16*y,"I");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(21*x,16*y,"J");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(23*x,16*y,"K");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(25*x,16*y,"L");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(27*x,16*y,"M");
Pause(30*x,24*y);
setcolor(backcolor);
```

```
    bar(29*x,23*y,70*x,49*y/2);
    setcolor(forecolor);
    /*******************************************************************/
    outtextxy(29*x,16*y,"N");
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,70*x,49*y/2);
    setcolor(forecolor);
    /*******************************************************************/
    outtextxy(31*x,16*y,"O");
    /*******************************************************************/
    setlinestyle(3,0,1);
    moveto(44*x,19*y/2); lineto(89*x,19*y/2);
    setlinestyle(0,0,3);
    outtextxy(44*x,10*y,"As you all see, the inorder listing is");
    outtextxy(44*x,11*y,"in alphabetical order. This way we have");
    outtextxy(44*x,12*y,"showed you one of the posible and easi-");
    outtextxy(44*x,13*y,"est ways to implement sorting. But as ");
    outtextxy(44*x,14*y,"we said earlier, if only sorting is in");
    outtextxy(44*x,15*y,"question, there are better and easier ");
    outtextxy(44*x,16*y,"ways to implement sorting.");
    /*******************************************************************/
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM      : examp464.c
  AUTHOR        : Atilla BAKAN
  DATE          : Apr. 18, 1990
  REVISED       : Apr. 18, 1990

  DESCRIPTION : This routine draws the example graph for searching on a binary
                search tree.


  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                     /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                      /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/* Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);

/***********************************************************************/
/* graphic initialization variables                                    */
/***********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/***********************************************************************/
/* This function is used for including drivers to the executable code  */
/***********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/**********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }
```

```
/***********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
   }
   switch (ch)       {
    case 'y': closegraph();
         videoinit();
         exit(0);
         break;
    case 'Y': closegraph();
         videoinit();
         exit(0);
         break;
    case 'n': setcolor(backcolor);
         bar(4*x/3,23*y,30*x,97*y/4);
         bar(31*x,23*y,69*x,97*y/4);
         setcolor(forecolor);
         break;
    case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}
/**********************************************************************/
/* This function sets the text default values                          */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/**********************************************************************/
/* main routine, calls exer routine                                    */
/**********************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates searching in a binary search tree.       */
/*********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-6-4");
  outtextxy(2*x,2*y,"On this binary search tree we will first search for A, then L.");
  pieslice(17*x,5*y,0,359,2);  /* K */
  pieslice(12*x,7*y,0,359,2);  /* B */
  pieslice(22*x,7*y,0,359,2);  /* T */
  moveto(12*x,7*y); lineto(17*x,5*y); lineto(22*x,7*y);
  outtextxy(17*x,9*y/2,"K");
  outtextxy(10*x,7*y,"B");
  outtextxy(24*x,7*y,"T");
  pieslice(17*x,9*y,0,359,2);  /* L */
  pieslice(27*x,9*y,0,359,2);  /* Z */
  moveto(17*x,9*y); lineto(22*x,7*y); lineto(27*x,9*y);
  outtextxy(17*x,19*y/2,"L");
  outtextxy(27*x,19*y/2,"Z");
  outtextxy(44*x,5*y,"Notes");
  moveto(43*x,11*y/2); lineto(89*x,11*y/2);
  outtextxy(3*x,29*y/2,"The nodes that we searched so far");
  moveto(2*x,15*y); lineto(40*x,15*y);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  outtextxy(44*x,6*y,"The search starts at the root, K. Since");
  outtextxy(44*x,7*y,"A < K, the search moves down the left");
  outtextxy(44*x,8*y,"subtree of K.");
  outtextxy(3*x,16*y,"K");
```

```
setlinestyle(3,0,1);
moveto(44*x,17*y/2); lineto(89*x,17*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,9*y,"We now at the left subtree(B) of root");
outtextxy(44*x,10*y,"K. Since A < B, the search would move");
outtextxy(44*x,11*y,"down to the left subtree of B.");
outtextxy(5*x,16*y,"B");
setlinestyle(3,0,1);
moveto(44*x,23*y/2); lineto(89*x,23*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,12*y,"However, B has no left subtree, so the");
outtextxy(44*x,13*y,"search terminates, with the item not");
outtextxy(44*x,14*y,"found.");
outtextxy(7*x,16*y,"Item not found!");
setlinestyle(3,0,1);
moveto(44*x,17*y/2); lineto(89*x,17*y/2);
setlinestyle(0,0,3);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
bar(43*x,23*y/4,179*x/2,49*y/2);
bar(3*x,61*y/4,43*x,20*y);
setcolor(forecolor);
/*********************************************************************/
outtextxy(44*x,6*y,"Now consider searching for the key L.");
outtextxy(44*x,7*y,"Since L > K, the search moves down the");
```

```
   outtextxy(44*x,8*y,"right subtree of K.");
   outtextxy(3*x,16*y,"K");
   setlinestyle(3,0,1);
   moveto(44*x,17*y/2); lineto(89*x,17*y/2);
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,70*x,49*y/2);
   setcolor(forecolor);
/*******************************************************************/
   outtextxy(44*x,9*y,"Now we are at the right subtree(T) of");
   outtextxy(44*x,10*y,"the root K. L < T, so the next move");
   outtextxy(44*x,11*y,"is down to the left subtree of T.");
   outtextxy(5*x,16*y,"T");
   setlinestyle(3,0,1);
   moveto(44*x,23*y/2); lineto(89*x,23*y/2);
   setlinestyle(0,0,3);
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,70*x,49*y/2);
   setcolor(forecolor);
/*******************************************************************/
   outtextxy(44*x,12*y,"Now we are at the left subtree(L) of T.");
   outtextxy(44*x,13*y,"Since this is the same as the search");
   outtextxy(44*x,14*y,"key, the search terminates, with the");
   outtextxy(44*x,15*y,"item found.");
   outtextxy(7*x,16*y,"L (Item found)");
   setlinestyle(3,0,1);
   moveto(44*x,31*y/2); lineto(89*x,31*y/2);
   setlinestyle(0,0,3);
   outtextxy(44*x,16*y,"We hope that you got the idea.");
/*******************************************************************/
   Pause(30*x,24*y);
   closegraph();
   videoinit();
}
```

```
/* PROGRAM  : q461.c
  AUTHOR     : Atilla BAKAN
  DATE       : Apr. 4, 1990
  REVISED    : Apr. 18, 1990

  DESCRIPTION : This program contains the first exercise about the
                binary search trees.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                       /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                        /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions    */
static void exer            (void);
static void show_alg        (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;



/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/************************************************************************/
/* This function is used for including drivers to the executable code          */
/************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/************************************************************************/
/* This fuction initializes the necessary graphical routines                  */
/************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**********************************************************************/
  settext();
  /**********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/****************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)         {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/*********************************************************************/
/* This function sets the text default values                      */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen          */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine, calls exer routine                               */
/*********************************************************************/
void main()
{
  exer();
}

/*********************************************************************/
/* This routine  asks the question, then depending on the user's answer     */
/* makes necessary explanations                                      */
/*********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  1");
   /*****************************************************************/
   outtextxy(5*x,2*y,"Construct a binary search tree for the following sequence of let-
                              ters.");
   outtextxy(10*x,3*y,"N, O, F, T, D, J, I, B, L, Q, U, K, P, A, V, S, G,");
/*************************************************************************/
while (in_the_exercise == 1) {
 outtextxy(15*x,14*y,"Choose one of the following, if you need :");
 outtextxy(15*x,15*y,"    a) I want to see the algorithm again.");
 outtextxy(15*x,16*y,"    b) I'm done, I want to compare my solution with yours.");
 outtextxy(15*x,17*y,"    c) I want to see step by step solution.");
 outtextxy(15*x,18*y,"    d) This is enough for me, I want to exit.");
 outtextxy(15*x,19*y,"Enter your choice here --->");
 Ch = getch ();
 if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
    outtextxy(48*x,19*y,"   Please type a, b, c or d");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
    setcolor(backcolor);
    bar(50*x,37*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
    switch (Ch)         {
    case 'a': outtextxy(47*x,19*y,"a");
      outtextxy(52*x,19*y,"You want to see the algorithm ");
      outtextxy(52*x,20*y,"again. Press any key to continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,37*y/2,179*x/2,21*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      show_alg();
      break;
   case 'b': outtextxy(47*x,19*y,"b");
      outtextxy(52*x,19*y,"You want to compare your solu-");
      outtextxy(52*x,20*y,"tion with ours. So press any  ");
      outtextxy(52*x,21*y,"key to see it.");
      Pause(30*x,24*y);
```

```
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        compare_solutions();
        break;
      case 'c': outtextxy(47*x,19*y,"c");
        outtextxy(52*x,19*y,"You want to see step by step");
        outtextxy(52*x,20*y,"solution. So press any key to ");
        outtextxy(52*x,21*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
     }
  }
  closegraph();
  videoinit();
}
```

```
/************************************************************************/
/* This routine gives preorder traversal of a binary tree algorithm         */
/************************************************************************/
static void show_alg(void)
{
  outtextxy(15*x,5*y,"BINARY SEARCH TREE CONSTRUCTION ALGO-
                         RITHM");
  outtextxy(2*x,7*y,"Step 1 (start). Construct a root and label it A1. If n = 1, we are
                        done;");
  outtextxy(2*x,8*y,"otherwise, let V = A1 and k = 2.");
  outtextxy(2*x,10*y,"Step 2 (if smaller, go left). If V <= A1, go to  Step 3. Other-
                         wise, we have");
  outtextxy(2*x,11*y,"Ak <= V.");
  outtextxy(2*x,13*y,"  (a) If V has no left child, construct a left child L for V and la-
                        bel L ");
  outtextxy(2*x,14*y,"with Ak. If k = n, we are done; otherwise, increase k by 1, set
                        V = A1, and ");
  outtextxy(2*x,15*y,"go to Step 2.");
  outtextxy(2*x,16*y,"  (b) Otherwise, if V has a left child L, set V = L, and go to
                         Step 2.");
  outtextxy(2*x,18*y,"Step 3 (if larger, go right). We have V <= Ak .");
  outtextxy(2*x,19*y,"  (a) If V has no right child, construct a right child R for V and
                          label R ");
  outtextxy(2*x,20*y,"with Ak. If k = n, we are done; otherwise, increase k by 1, set
                         V = A1, and");
  outtextxy(2*x,21*y,"go to Step 2.");
  outtextxy(2*x,22*y,"  (b) Otherwise, if V has a right child R, set V = R, and go to
                         Step 2.");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(2*x,7*y/2,179*x/2,49*y/2);
  setcolor(forecolor);
}
```

```
/***********************************************************************/
/* This routine gives the solution to the exercise to be compared.     */
/***********************************************************************/
static void compare_solutions(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,7*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
   /********************************************************************/
   pieslice(40*x,5*y,0,359,2);
   pieslice(35*x,7*y,0,359,2);
   pieslice(45*x,7*y,0,359,2);
   pieslice(30*x,9*y,0,359,2);
   pieslice(38*x,9*y,0,359,2);
   pieslice(50*x,9*y,0,359,2);
   pieslice(25*x,11*y,0,359,2);
   pieslice(35*x,11*y,0,359,2);
   pieslice(42*x,11*y,0,359,2);
   pieslice(46*x,11*y,0,359,2);
   pieslice(55*x,11*y,0,359,2);
   pieslice(20*x,13*y,0,359,2);
   pieslice(30*x,13*y,0,359,2);
   pieslice(38*x,13*y,0,359,2);
   pieslice(42*x,13*y,0,359,2);
   pieslice(50*x,13*y,0,359,2);
   pieslice(60*x,13*y,0,359,2);
   moveto(20*x,13*y); lineto(25*x,11*y);
   lineto(30*x,9*y);  lineto(35*x,7*y);
   lineto(40*x,5*y);  lineto(45*x,7*y);
   lineto(50*x,9*y);  lineto(55*x,11*y);
   lineto(60*x,13*y);
   moveto(30*x,13*y); lineto(35*x,11*y);
   lineto(38*x,9*y);  lineto(42*x,11*y);
   lineto(38*x,13*y);
   moveto(35*x,7*y); lineto(38*x,9*y);
   moveto(42*x,13*y); lineto(46*x,11*y);
```

```
    lineto(50*x,13*y);
    moveto(46*x,11*y); lineto(50*x,9*y);
    outtextxy(79*x/2,9*y/2,"N");
    outtextxy(33*x,7*y,"F");
    outtextxy(46*x,7*y,"O");
    outtextxy(28*x,9*y,"D");
    outtextxy(39*x,9*y,"J");
    outtextxy(51*x,9*y,"T");
    outtextxy(23*x,11*y,"B");
    outtextxy(33*x,11*y,"I");
    outtextxy(43*x,11*y,"L");
    outtextxy(47*x,11*y,"Q");
    outtextxy(56*x,11*y,"U");
    outtextxy(20*x,27*y/2,"A");
    outtextxy(30*x,27*y/2,"G");
    outtextxy(38*x,27*y/2,"K");
    outtextxy(42*x,27*y/2,"P");
    outtextxy(50*x,27*y/2,"S");
    outtextxy(60*x,27*y/2,"V");
    /*****************************************************************/
    Pause(30*x,24*y);
    setcolor(backcolor);          /* Clean the game field again */
    bar(2*x,7*y/2,179*x/2,49*y/2);
    setcolor(forecolor);
}
```

```
/*************************************************************************/
/* This routine gives the step by step solution to the exerc. e           */
/*************************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);         /* Clean the game field */
  bar(2*x,7*y/2 179*x/2,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  pieslice(40*x,5*y,0,359,2);
  outtextxy(79*x/2,9*y/2,"N");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  pieslice(45*x,7*y,0,359,2);
  outtextxy(46*x,7*y,"O");
  moveto(40*x,5*y); lineto(45*x,7*y);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  pieslice(35*x,7*y,0,359,2);
  outtextxy(33*x,7*y,"F");
  moveto(40*x,5*y); lineto(35*x,7*y);
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*********************************************************************/
  pieslice(50*x,9*y,0,359,2);
  outtextxy(51*x,9*y,"T");
  moveto(50*x,9*y); lineto(45*x,7*y);
  Pause(30*x,24*y);
```

```
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
pieslice(30*x,9*y,0,359,2);
outtextxy(28*x,9*y,"D");
moveto(30*x,9*y); lineto(35*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
pieslice(38*x,9*y,0,359,2);
outtextxy(39*x,9*y,"J");
moveto(38*x,9*y); lineto(35*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
pieslice(35*x,11*y,0,359,2);
outtextxy(33*x,11*y,"I");
moveto(35*x,11*y); lineto(38*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
pieslice(25*x,11*y,0,359,2);
outtextxy(23*x,11*y,"B");
moveto(25*x,11*y); lineto(30*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/********************************************************************/
```

```
pieslice(42*x,11*y,0,359,2);
outtextxy(43*x,11*y,"L");
moveto(42*x,11*y); lineto(38*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(46*x,11*y,0,359,2);
outtextxy(47*x,11*y,"Q");
moveto(46*x,11*y); lineto(50*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(55*x,11*y,0,359,2);
outtextxy(56*x,11*y,"U");
moveto(55*x,11*y); lineto(50*x,9*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(38*x,13*y,0,359,2);
outtextxy(38*x,27*y/2,"K");
moveto(42*x,11*y); lineto(38*x,13*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(42*x,13*y,0,359,2);
outtextxy(42*x,27*y/2,"P");
moveto(46*x,11*y); lineto(42*x,13*y);
Pause(30*x,24*y);
```

```
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(20*x,13*y,0,359,2);
outtextxy(20*x,27*y/2,"A");
moveto(25*x,11*y);  lineto(20*x,13*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(60*x,13*y,0,359,2);
outtextxy(60*x,27*y/2,"V");
moveto(55*x,11*y);  lineto(60*x,13*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(50*x,13*y,0,359,2);
outtextxy(50*x,27*y/2,"S");
moveto(46*x,11*y);  lineto(50*x,13*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(30*x,13*y,0,359,2);
outtextxy(30*x,27*y/2,"G");
moveto(35*x,11*y);  lineto(30*x,13*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
setcolor(backcolor);          /* Clean the game field */
```

```
   bar(2*x,7*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
}
/*************************************************************************/
static void confirm_e⁻·'(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       outtextxy(53*x,23*y," Please type y or n");
       ch = getch ();
       if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
       setcolor(backcolor);
       bar(50*x,22*y,179*x/2,49*y/2);
       setcolor(forecolor);
   }
    switch (ch)          {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;
    case 'n': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;
    case 'N': setcolor(backcolor);
           bar(46*x,37*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;
    default : break;
    }
}
```

```
/* PROGRAM   : q462.c
  AUTHOR      : Atilla BAKAN
  DATE        : Apr. 4, 1990
  REVISED     : Apr. 18, 1990

  DESCRIPTION : This program contains the second exercise about the
                binary search trees.

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                    /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk          find_t
  #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
  #define ffblk          FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions     */
static void exer             (void);
static void show_alg          (void);
static void step_solution     (void);
static void compare_solutions (void);
static void confirm_exit      (void);


/**********************************************************************/
/* miscellaneous global variables                                    */
/**********************************************************************/
 int in_the_exercise = 1;


/**********************************************************************/
/* graphic initialization variables                                  */
/**********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/**********************************************************************/
/* This function is used for including drivers to the executable code      */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/**********************************************************************/
/* This fuction initializes the necessary graphical routines              */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*********************************************************************/
  settext();
  /*********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
     setfillstyle(SOLID_FILL,BLACK);
     backcolor = BLACK;
     quitcolor = WHITE;
     }
   else {
     setfillstyle(SOLID_FILL,BLUE);
     backcolor = BLUE;
     quitcolor = RED;
     }
  forecolor = WHITE;
 }

/**************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL); /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
     case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
     case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}



/************************************************************************/
/* This function sets the text default values                          */
/************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen           */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine, calls exer routine                                  */
/*********************************************************************/
void main()
{
  exer();
}


/*********************************************************************/
/* This routine  asks the question, then depending on the user's answer */
/* makes necessary explanations                                      */
/*********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  2");
   /*****************************************************************/
   outtextxy(5*x,2*y,"Construct a binary search tree for the following sequence of
                              names.");
outtextxy(5*x,3*y,"Mary, Jack, John, Chris, Natalie, Denise, Vanna, Tom, Queen,
                    Tony, Mona");
outtextxy(5*x,4*y,"Bill, Jean, Zamphir");
/*****************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"     a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"     b) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,17*y,"     c) I want to see step by step solution.");
outtextxy(15*x,18*y,"     d) This is enough for me, I want to exit.");
outtextxy(15*x,19*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
 while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,19*y,"   Please type a, b, c or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd')) {
   setcolor(backcolor);
   bar(50*x,37*y/2,88*x,20*y);
   setcolor(forecolor);
   }
  }
   switch (Ch)          {
   case 'a': outtextxy(47*x,19*y,"a");
     outtextxy(52*x,19*y,"You want to see the algorithm ");
     outtextxy(52*x,20*y,"again. Press any key to continue.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,37*y/2,179*x/2,21*y);
     bar(2*x,13*y,179*x/2,49*y/2);
     setcolor(forecolor);
     show_alg();
     break;
  case 'b': outtextxy(47*x,19*y,"b");
     outtextxy(52*x,19*y,"You want to compare your solu-");
```

```
        outtextxy(52*x,20*y,"tion with ours. So press any  ");
        outtextxy(52*x,21*y,"key to see it.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        compare_solutions();
        break;
      case 'c': outtextxy(47*x,19*y,"c");
        outtextxy(52*x,19*y,"You want to see step by step");
        outtextxy(52*x,20*y,"solution. So press any key to ");
        outtextxy(52*x,21*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,37*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'd': outtextxy(47*x,19*y,"d");
        confirm_exit();
        break;
      default  : break;
    }
  }
  closegraph();
  videoinit();
}
```

```
/***********************************************************************/
/* This routine gives preorder traversal of a binary tree algorithm               */
/***********************************************************************/
static void show_alg(void)
{
   outtextxy(15*x,5*y,"BINARY SEARCH TREE CONSTRUCTION ALGO-
                              RITHM");
   outtextxy(2*x,7*y,"Step 1 (start). Construct a root and label it A1. If n = 1, we are
                             done;");
   outtextxy(2*x,8*y,"otherwise, let V = A1 and k = 2.");
   outtextxy(2*x,10*y,"Step 2 (if smaller, go left). If V <= A1, go to  Step 3. Other-
                             wise, we have");
   outtextxy(2*x,11*y,"Ak <= V.");
   outtextxy(2*x,12*y,"  (a) If V has no left child, construct a left child L for V and la-
                              bel L ");
   outtextxy(2*x,13*y,"with Ak. If k = n, we are done; otherwise, increase k by 1, set
                              V = A1, and ");
   outtextxy(2*x,14*y,"go to Step 2.");
   outtextxy(2*x,15*y,"  (b) Otherwise, if V has a left child L, set V = L, and go to
                              Step 2.");
   outtextxy(2*x,17*y,"Step 3 (if larger, go right). We have V <= Ak .");
   outtextxy(2*x,18*y,"  (a) If V has no right child, construct a right child R for V and
                              label R ");
   outtextxy(2*x,19*y,"with Ak. If k = n, we are done; otherwise, increase k by 1, set
                              V = A1, and");
   outtextxy(2*x,20*y,"go to Step 2.");
   outtextxy(2*x,21*y,"  (b) Otherwise, if V has a right child R, set V = R, and go to
                              Step 2.");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(2*x,9*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
}
```

```
/***********************************************************************/
/* This routine gives the solution to the exercise to be compared.     */
/***********************************************************************/
static void compare_solutions(void)
{
   setcolor(backcolor);         /* Clean the game field */
   bar(2*x,9*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
   /*******************************************************************/
   pieslice(40*x,5*y,0,359,2);       /* Mary    */
   pieslice(25*x,7*y,0,359,2);       /* John    */
   pieslice(55*x,7*y,0,359,2);       /* Natalie */
   pieslice(20*x,9*y,0,359,2);       /* Jack    */
   pieslice(50*x,9*y,0,359,2);       /* Mona    */
   pieslice(60*x,9*y,0,359,2);       /* Vanna   */
   pieslice(15*x,11*y,0,359,2);      /* Chris   */
   pieslice(25*x,11*y,0,359,2);      /* Jean    */
   pieslice(55*x,11*y,0,359,2);      /* Tom     */
   pieslice(65*x,11*y,0,359,2);      /* Zamphir */
   pieslice(10*x,13*y,0,359,2);      /* Bill    */
   pieslice(20*x,13*y,0,359,2);      /* Denise  */
   pieslice(50*x,13*y,0,359,2);      /* Queen   */
   pieslice(60*x,13*y,0,359,2);      /* Tony    */
   moveto(10*x,13*y); lineto(15*x,11*y);
   lineto(20*x,9*y);  lineto(25*x,7*y);
   lineto(40*x,5*y);  lineto(55*x,7*y);
   lineto(60*x,9*y);  lineto(65*x,11*y);
   moveto(20*x,13*y); lineto(15*x,11*y);
   moveto(25*x,11*y); lineto(20*x,9*y);
   moveto(55*x,7*y);  lineto(50*x,9*y);
   moveto(50*x,13*y); lineto(55*x,11*y);
   lineto(60*x,9*y);
   moveto(60*x,13*y); lineto(55*x,11*y);
   outtextxy(37*x,19*y/4,"Mary");
   outtextxy(19*x,7*y,"John");
   outtextxy(56*x,7*y,"Natalie");
   outtextxy(14*x,9*y,"Jack");
   outtextxy(47*x,19*y/2,"Mona");
   outtextxy(61*x,9*y,"Vanna");
   outtextxy(8*x,11*y,"Chris");
   outtextxy(22*x,23*y/2,"Jean");
   outtextxy(50*x,11*y,"Tom");
   outtextxy(60*x,23*y/2,"Zamphir");
   outtextxy(7*x,27*y/2,"Bill");
   outtextxy(16*x,27*y/2,"Denise");
   outtextxy(47*x,27*y/2,"Queen");
   outtextxy(58*x,27*y/2,"Tony");
   /*****************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field again */
   bar(2*x,9*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
}


/******************************************************************/
/* This routine gives the step by step solution to the exercise        */
/******************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,9*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   pieslice(40*x,5*y,0,359,2);      /* Mary    */
   outtextxy(37*x,19*y/4,"Mary");
   Pause(30*x,24*y);
   setcolor(backcolor);
   bar(29*x,23*y,50*x,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   pieslice(25*x,7*y,0,359,2);      /* John    */
```

```
outtextxy(19*x,7*y,"John");
moveto(40*x,5*y); lineto(25*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(20*x,9*y,0,359,2);     /* Jack   */
outtextxy(14*x,9*y,"Jack");
moveto(20*x,9*y); lineto(25*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(15*x,11*y,0,359,2);     /* Chris  */
outtextxy(8*x,11*y,"Chris");
moveto(20*x,9*y); lineto(15*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(55*x,7*y,0,359,2);     /* Natalie */
outtextxy(56*x,7*y,"Natalie");
moveto(40*x,5*y); lineto(55*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(20*x,13*y,0,359,2);     /* Denise  */
outtextxy(17*x,27*y/2,"Denise");
moveto(20*x,13*y); lineto(15*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
```

```
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***************************************************************/
pieslice(60*x,9*y,0,359,2);     /* Vanna  */
outtextxy(61*x,9*y,"Vanna");
moveto(60*x,9*y); lineto(55*x,7*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***************************************************************/
pieslice(55*x,11*y,0,359,2);    /* Tom    */
outtextxy(50*x,11*y,"Tom");
moveto(60*x,9*y); lineto(55*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***************************************************************/
pieslice(50*x,13*y,0,359,2);    /* Queen  */
outtextxy(47*x,27*y/2,"Queen");
moveto(50*x,13*y); lineto(55*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***************************************************************/
pieslice(60*x,13*y,0,359,2);    /* Tony   */
outtextxy(58*x,27*y/2,"Tony");
moveto(60*x,13*y); lineto(55*x,11*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/***************************************************************/
pieslice(50*x,9*y,0,359,2);     /* Mona   */
```

```
    outtextxy(47*x,19*y/2,"Mona");
    moveto(50*x,9*y); lineto(55*x,7*y);
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /*******************************************************************/
    pieslice(10*x,13*y,0,359,2);      /* Bill    */
    outtextxy(7*x,27*y/2,"Bill");
    moveto(10*x,13*y); lineto(15*x,11*y);
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /*******************************************************************/
    pieslice(25*x,11*y,0,359,2);      /* Jean    */
    outtextxy(22*x,23*y/2,"Jean");
    moveto(20*x,9*y); lineto(25*x,11*y);
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /*******************************************************************/
    pieslice(65*x,11*y,0,359,2);      /* Zamphir */
    outtextxy(60*x,23*y/2,"Zamphir");
    moveto(60*x,9*y); lineto(65*x,11*y);
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(29*x,23*y,50*x,49*y/2);
    setcolor(forecolor);
    /*******************************************************************/
    setcolor(backcolor);          /* Clean the game field */
    bar(2*x,9*y/2,179*x/2,49*y/2);
    setcolor(forecolor);
}
```

```
/************************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,19*y,"You wanted to exit. ");
  outtextxy(52*x,20*y,"Are you sure ? ");
  outtextxy(52*x,21*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,23*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
   switch (ch)         {
    case 'y': in_the_exercise = 0;
          break;
    case 'Y': in_the_exercise = 0;
          break;

    case 'n': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    case 'N': setcolor(backcolor);
          bar(46*x,37*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    default : break;
    }
}
```

```
/* PROGRAM   : q463.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 4, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This program contains the third exercise about the binary
                 search trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                      /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;



/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code          */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines                    */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /******************************************************************/
  settext();
  /******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/*****************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)          {
     case 'y': closegraph();
```

```
                videoinit();
                exit(0);
                break;
      case 'Y': closegraph();
                videoinit();
                exit(0);
                break;
      case 'n': setcolor(backcolor);
                bar(4*x/3,23*y,30*x,97*y/4);
                bar(31*x,23*y,69*x,97*y/4);
                setcolor(forecolor);
                break;
      case 'N': setcolor(backcolor);
                bar(4*x/3,23*y,30*x,97*y/4);
                bar(31*x,23*y,69*x,97*y/4);
                setcolor(forecolor);
                break;
      default : break;
      }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);      /* restore any hidden hot keys */
  }




/*********************************************************************/
/* This function sets the text default values                       */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen           */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine, calls exer routine                                  */
/*********************************************************************/
void main()
{
  exer();
}

/*********************************************************************/
/* This routine  asks the question, then depending on the user's answer */
/* makes necessary explanations                                      */
/*********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  3");
   outtextxy(20*x,2*y,"Consider the following binary tree.");
   /*****************************************************************/
   pieslice(40*x,3*y,0,359,2);
   pieslice(35*x,5*y,0,359,2);
pieslice(45*x,5*y,0,359,2);
pieslice(30*x,7*y,0,359,2);
pieslice(38*x,7*y,0,359,2);
pieslice(50*x,7*y,0,359,2);
pieslice(25*x,9*y,0,359,2);
pieslice(35*x,9*y,0,359,2);
pieslice(42*x,9*y,0,359,2);
pieslice(46*x,9*y,0,359,2);
pieslice(55*x,9*y,0,359,2);
pieslice(20*x,11*y,0,359,2);
pieslice(30*x,11*y,0,359,2);
pieslice(38*x,11*y,0,359,2);
pieslice(42*x,11*y,0,359,2);
pieslice(50*x,11*y,0,359,2);
pieslice(60*x,11*y,0,359,2);
moveto(20*x,11*y); lineto(25*x,9*y);
lineto(30*x,7*y);  lineto(35*x,5*y);
lineto(40*x,3*y);  lineto(45*x,5*y);
lineto(50*x,7*y);  lineto(55*x,9*y);
lineto(60*x,11*y);
moveto(30*x,11*y); lineto(35*x,9*y);
lineto(38*x,7*y);  lineto(42*x,9*y);
lineto(38*x,11*y);
moveto(35*x,5*y); lineto(38*x,7*y);
moveto(42*x,11*y); lineto(46*x,9*y);
lineto(50*x,11*y);
moveto(46*x,9*y); lineto(50*x,7*y);
outtextxy(41*x,3*y,"N");
outtextxy(33*x,5*y,"F");
outtextxy(46*x,5*y,"O");
outtextxy(28*x,7*y,"D");
outtextxy(39*x,7*y,"J");
outtextxy(51*x,7*y,"T");
outtextxy(23*x,9*y,"B");
outtextxy(33*x,9*y,"I");
outtextxy(43*x,9*y,"L");
outtextxy(47*x,9*y,"Q");
outtextxy(56*x,9*y,"U");
outtextxy(20*x,23*y/2,"A");
outtextxy(30*x,23*y/2,"G");
outtextxy(38*x,23*y/2,"K");
outtextxy(42*x,23*y/2,"P");
outtextxy(50*x,23*y/2,"S");
outtextxy(60*x,23*y/2,"V");
/*******************************************************************/
outtextxy(18*x,13*y,"Which one of the following statements is  false ?");
outtextxy(20*x,15*y,"a)  Search pattern for C is 'N, F, D, B, Item not found'.");
outtextxy(20*x,16*y,"b)  Search pattern for Q is 'N, O, T, Q, Item found'.");
outtextxy(20*x,17*y,"c)  M would be added to the list as right child of K.");
outtextxy(20*x,18*y,"d)  R would be added to the list as left child of S.");
outtextxy(18*x,20*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,20*y,"    Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(50*x.19*y,88*x,21*y);
   setcolor(forecolor);
 }
 switch (Ch)         {
  case 'a': outtextxy(50*x,20*y,"a");
      outtextxy(55*x,20*y,"No, the statement is true!");
      outtextxy(55*x,21*y,"If C were in the tree, it");
      outtextxy(55*x,22*y,"would be B's right child.");
      outtextxy(55*x,23*y,"As you see, this not the case.");
      outtextxy(55*x,24*y,"The answer is 'c'.");
      break;
  case 'b': outtextxy(50*x,20*y,"b");
      outtextxy(55*x,20*y,"No, the statement is true!");
```

```
        outtextxy(55*x,21*y,"Q is in the tree and since");
        outtextxy(55*x,22*y,"Q>N, Q>O, Q<T, and Q=Q the");
        outtextxy(55*x,23*y,"search pattern is correct.");
        outtextxy(55*x,24*y,"The answer is 'c'.");
        break;
    case 'c': outtextxy(50*x,20*y,"c");
        outtextxy(55*x,20*y,"You are right. Congratulations");
        break;
    case 'd': outtextxy(50*x,20*y,"d");
        outtextxy(55*x,20*y,"No, the statement is true!");
        outtextxy(55*x,21*y,"Because, R>N, R>O, R<T, R>Q");
        outtextxy(55*x,22*y,"and R<S.");
        outtextxy(55*x,23*y,"The answer is 'c'.");
        break;
    default  : break;
    }
    Pause(15*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : q464.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 8, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This program contains the fourth exercise about the
                 binary search trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                    /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit     (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;



/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code   */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**********************************************************************/
  settext();
  /**********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
      ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
        setfillstyle(SOLID_FILL,BLACK);
        backcolor = BLACK;
        quitcolor = WHITE;
        }
    else {
        setfillstyle(SOLID_FILL,BLUE);
        backcolor = BLUE;
        quitcolor = RED;
        }
   forecolor = WHITE;
  }

/***********************************************************************/
static void confirm_graph_exit(void)
{
    struct _onkey_t *kblist;
    char ch;

    setcolor(backcolor);
    bar(3*x/2,23*y,179*x/2,97*y/4);
    setcolor(quitcolor);
    kblist=chgonkey(NULL);  /* hide any existing hot keys */
    if(_mouse&MS_CURS) mshidecur();
    outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
    ch = getch ();
    while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
        outtextxy(32*x,24*y," Please type y or n");
        ch = getch ();
        if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
        setcolor(backcolor);
        bar(31*x,23*y,69*x,97*y/4);
        setcolor(quitcolor);
     }
     switch (ch)         {
      case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
      case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
      case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);     /* restore any hidden hot keys */
}



/**************************************************************************/
/* This function sets the text default values                            */
/**************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen           */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/*********************************************************************/
/* main routine, calls exer routine                                  */
/*********************************************************************/
void main()
{
  exer();
}

/*********************************************************************/
/* This routine  asks the question, then depending on the user's answer  */
/* makes necessary explanations                                      */
/*********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  4");
   /*****************************************************************/
   pieslice(10*x,5*y,0,359,2);          /* a */
   pieslice(7*x,7*y,0,359,2);
   pieslice(13*x,7*y,0,359,2);
pieslice(3*x,9*y,0,359,2);
pieslice(17*x,9*y,0,359,2);
pieslice(7*x,11*y,0,359,2);
pieslice(13*x,11*y,0,359,2);
moveto(13*x,11*y); lineto(17*x,9*y);
lineto(13*x,7*y);  lineto(10*x,5*y);
lineto(7*x,7*y);   lineto(3*x,9*y);
lineto(7*x,11*y);
outtextxy(10*x,9*y/2,"A");
outtextxy(5*x,7*y,"B");
outtextxy(14*x,7*y,"C");
outtextxy(3*x/2,9*y,"D");
outtextxy(18*x,9*y,"E");
outtextxy(7*x,23*y/2,"F");
outtextxy(13*x,23*y/2,"G");
outtextxy(10*x,13*y,"(a)");
/*********************************************************************/
pieslice(24*x,5*y,0,359,2);          /* b */
pieslice(26*x,6*y,0,359,2);
pieslice(28*x,7*y,0,359,2);
pieslice(30*x,8*y,0,359,2);
pieslice(32*x,9*y,0,359,2);
pieslice(34*x,10*y,0,359,2);
pieslice(36*x,11*y,0,359,2);
moveto(24*x,5*y); lineto(26*x,6*y);
lineto(28*x,7*y); lineto(30*x,8*y);
lineto(32*x,9*y); lineto(34*x,10*y);
lineto(36*x,11*y);
outtextxy(25*x,5*y,"A");
outtextxy(27*x,6*y,"B");
outtextxy(29*x,7*y,"C");
outtextxy(31*x,8*y,"D");
outtextxy(33*x,9*y,"E");
outtextxy(35*x,10*y,"F");
outtextxy(37*x,11*y,"G");
outtextxy(30*x,13*y,"(b)");
```

```
/*********************************************************************/
pieslice(44*x,11*y,0,359,2);          /* c */
pieslice(46*x,10*y,0,359,2);
pieslice(48*x,9*y,0,359,2);
pieslice(50*x,8*y,0,359,2);
pieslice(52*x,7*y,0,359,2);
pieslice(54*x,6*y,0,359,2);
pieslice(56*x,5*y,0,359,2);
moveto(44*x,11*y); lineto(46*x,10*y);
lineto(48*x,9*y); lineto(50*x,8*y);
lineto(52*x,7*y); lineto(54*x,6*y);
lineto(56*x,5*y);
outtextxy(42*x,11*y,"7");
outtextxy(44*x,10*y,"6");
outtextxy(46*x,9*y,"5");
outtextxy(48*x,8*y,"4");
outtextxy(50*x,7*y,"3");
outtextxy(52*x,6*y,"2");
outtextxy(54*x,5*y,"1");
outtextxy(50*x,13*y,"(c)");
/*********************************************************************/
pieslice(64*x,7*y,0,359,2);          /* d */
pieslice(68*x,5*y,0,359,2);
pieslice(72*x,7*y,0,359,2);
pieslice(68*x,9*y,0,359,2);
pieslice(76*x,9*y,0,359,2);
pieslice(72*x,11*y,0,359,2);
pieslice(80*x,11*y,0,359,2);
moveto(64*x,7*y); lineto(68*x,5*y);
lineto(72*x,7*y); lineto(76*x,9*y);
lineto(80*x,11*y);
moveto(72*x,7*y); lineto(68*x,9*y);
moveto(76*x,9*y); lineto(72*x,11*y);
outtextxy(68*x,9*y/2,"2");
outtextxy(62*x,7*y,"1");
outtextxy(73*x,7*y,"3");
outtextxy(66*x,9*y,"4");
outtextxy(77*x,9*y,"6");
outtextxy(72*x,23*y/2,"5");
outtextxy(80*x,23*y/2,"7");
outtextxy(70*x,13*y,"(d)");
/******************************************************************/
outtextxy(10*x,2*y,"Which one of the following graphs is a binary search tree ?");
outtextxy(10*x,18*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,18*y,"   Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(50*x,17*y,88*x,19*y);
   setcolor(forecolor);
 }
 switch (Ch)        {
  case 'a': outtextxy(42*x,18*y,"a");
       outtextxy(47*x,18*y,"No. This is not a binary search");
       outtextxy(47*x,19*y,"tree, because the order of the");
       outtextxy(47*x,20*y,"elements is not proper.For example");
       outtextxy(47*x,21*y,"take B, since B>A, it must be on the");
       outtextxy(47*x,22*y,"right subtree of the root A. There");
       outtextxy(47*x,23*y,"are other examples, as well.");
       outtextxy(47*x,24*y,"The answer is 'b'");
       Pause(8*x,24*y);
       break;
  case 'b': outtextxy(42*x,18*y,"b");
       outtextxy(47*x,18*y,"That's right! Congratulations");
       Pause(30*x,24*y);
       break;
  case 'c': outtextxy(42*x,18*y,"c");
       outtextxy(47*x,18*y,"No. This is not a binary search");
```

```
        outtextxy(47*x,19*y,"tree, because the order of the");
        outtextxy(47*x,20*y,"elements is not proper. Everything");
        outtextxy(47*x,21*y,"on the left subtree of 1 supposed");
        outtextxy(47*x,22*y,"to be on its right subtree since");
        outtextxy(47*x,23*y,"1 is less than all of them.");
        outtextxy(47*x,24*y,"The answer is 'b'");
        Pause(8*x,24*y);
        break;
    case 'd': outtextxy(42*x,18*y,"d");
        outtextxy(47*x,18*y,"No. This is not a binary search");
        outtextxy(47*x,19*y,"tree, because the order of the");
        outtextxy(47*x,20*y,"elements is not proper. Because");
        outtextxy(47*x,21*y,"since 4>3, it must be on the");
        outtextxy(47*x,22*y,"right subtree of the vertex 3.");
        outtextxy(47*x,23*y,"The answer is 'b'");
        Pause(30*x,24*y);
        break;
    default  : break;
    }
    closegraph();
    videoinit();
}
```

```
/* PROGRAM    : q465.c
   AUTHOR     :  Atilla BAKAN
   DATE       : Apr. 8, 1990
   REVISED    : Apr. 18, 1990

   DESCRIPTION : This program contains the fifth exercise about the
                 binary search trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name          name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions      */
static void exer              (void);
static void show_alg          (void);
static void step_solution     (void);
static void compare_solutions (void);
static void confirm_exit      (void);


/*********************************************************************/
/* miscellaneous global variables                                    */
/*********************************************************************/
 int in_the_exercise = 1;


/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*********************************************************************/
/* This function is used for including drivers to the executable code        */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*********************************************************************/
/* This fuction initializes the necessary graphical routines               */
/*********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*******************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*******************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*******************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*******************************************************************/
  settext();
  /*******************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/*****************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
                videoinit();
                exit(0);
                break;
      case 'Y': closegraph();
                videoinit();
                exit(0);
                break;
      case 'n': setcolor(backcolor);
                bar(4*x/3,23*y,30*x,97*y/4);
                bar(31*x,23*y,69*x,97*y/4);
                setcolor(forecolor);
                break;
      case 'N': setcolor(backcolor);
                bar(4*x/3,23*y,30*x,97*y/4);
                bar(31*x,23*y,69*x,97*y/4);
                setcolor(forecolor);
                break;
      default : break;
      }
    hidecur();
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);     /* restore any hidden hot keys */
}




/**********************************************************************/
/* This function sets the text default values                        */
/**********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/************************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }

/************************************************************************/
/* main routine,  calls exer routine                                     */
/************************************************************************/
void main()
{
  exer();
}


/************************************************************************/
/* This routine asks the question, then depending on the user's answer */
/* makes necessary explanations                                         */
/************************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  5");
   /*********************************************************************/
   outtextxy(2*x,2*y,"Give sorted list of the following sequence of names by using bi-
                           nary search");
outtextxy(2*x,3*y,"tree approach : Natalie, Jack, John, Vanna, Mary, Zamphir,
                        Chris, Denise,");
outtextxy(2*x,4*y,"            Tony, Quincy, Tom, Bill, Jean, Mona");
/*****************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, if you need :");
outtextxy(15*x,15*y,"   a) I want to see the algorithm again.");
outtextxy(15*x,16*y,"   b) I want to see the solution.");
outtextxy(15*x,17*y,"   c) This is enough for me, I want to exit.");
outtextxy(15*x,18*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c'))) {
    outtextxy(48*x,18*y,"   Please type a, b, or c");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c')) {
    setcolor(backcolor);
    bar(50*x,35*y/2,88*x,20*y);
    setcolor(forecolor);
     }
   }
    switch (Ch)         {
    case 'a': outtextxy(47*x,18*y,"a");
      outtextxy(52*x,18*y,"You want to see the algorithm ");
      outtextxy(52*x,19*y,"again. Press any key to continue.");
      Pause(30*x,24*y);
      setcolor(backcolor);
      bar(50*x,35*y/2,179*x/2,21*y);
      bar(2*x,13*y,179*x/2,49*y/2);
      setcolor(forecolor);
      show_alg();
      break;
   case 'b': outtextxy(47*x,18*y,"b");
      outtextxy(52*x,18*y,"You want to see the solution.");
      outtextxy(52*x,19*y,"Press any key to continue.");
```

```
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,35*y/2,179*x/2,22*y);
        bar(2*x,13*y,179*x/2,49*y/2);
        setcolor(forecolor);
        step_solution();
        break;
      case 'c': outtextxy(47*x,18*y,"c");
        confirm_exit();
        break;
      default  : break;
    }
  }
  closegraph();
  videoinit();
}


/**********************************************************************/
/* This routine gives preorder traversal of a binary tree algorithm             */
/**********************************************************************/
static void show_alg(void)
{
  outtextxy(2*x,5*y,"1. Apply the following (BFS) algorithm to construct binary
                        search tree.");
  outtextxy(3*x,6*y,"Step 1 (start). Construct a root and label it A1. If n = 1, we are
                        done;");
  outtextxy(3*x,7*y,"otherwise, let V = A1 and k = 2.");
  outtextxy(3*x,8*y,"Step 2 (if smaller, go left). If V <= A1, go to  Step 3. Other-
                        wise, we have");
  outtextxy(3*x,9*y,"Ak <= V.");
  outtextxy(3*x,10*y,"  (a) If V has no left child, construct a left child L for V and la-
                        bel L ");
  outtextxy(3*x,11*y,"with Ak. If k = n, we are done; otherwise, increase k by 1, set
                        V = A1, and ");
  outtextxy(3*x,12*y,"go to Step 2.");
```

```
    outtextxy(3*x,13*y,"  (b) Otherwise, if V has a left child L, set V = L, and go to
                        Step 2.");
    outtextxy(3*x,14*y,"Step 3 (if larger, go right). We have V <= Ak .");
    outtextxy(3*x,15*y,"  (a) If V has no right child, construct a right child R for V and
                        label R ");
    outtextxy(3*x,16*y,"with Ak. If k = n, we are done; otherwise, increase k by 1, set
                        V = A1, and");
    outtextxy(3*x,17*y,"go to Step 2.");
    outtextxy(3*x,18*y,"  (b) Otherwise, if V has a right child R, set V = R, and go to
                        Step 2.");
    outtextxy(2*x,20*y,"2. Apply following (inorder traversal) algorithm to get the
                        sorted list.");
    outtextxy(3*x,21*y,"Step 1 Go to the left subtree, if one  exists, do a preorder tra-
                        versal");
    outtextxy(3*x,22*y,"Step 2 Visit the root.");
    outtextxy(3*x,23*y,"Step 3 Go to the right subtree, if one exists, and do a preorder
                        traversal.");
    Pause(30*x,24*y);
    setcolor(backcolor);
    bar(2*x,9*y/2,179*x/2,49*y/2);
    setcolor(forecolor);
}
```

```
/**********************************************************************/
/* This routine gives the step by step solution to the exercise       */
/**********************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,9*y/2,179*x/2,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  pieslice(40*x,5*y,0,359,2);       /* Natalie */
  outtextxy(36*x,19*y/4,"Natalie");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  pieslice(25*x,7*y,0,359,2);       /* Jack    */
  moveto(40*x,5*y); lineto(25*x,7*y);
  outtextxy(19*x,7*y,"Jack");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  pieslice(30*x,9*y,0,359,2);       /* John    */
  moveto(25*x,7*y); lineto(30*x,9*y);
  outtextxy(31*x,9*y,"John");
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,50*x,49*y/2);
  setcolor(forecolor);
  /*******************************************************************/
  pieslice(55*x,7*y,0,359,2);       /* Vanna  */
  moveto(40*x,5*y); lineto(55*x,7*y);
  outtextxy(56*x,7*y,"Vanna");
  Pause(30*x,24*y);
```

```
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(35*x,11*y,0,359,2);     /* Mary    */
moveto(30*x,9*y);  lineto(35*x,11*y);
outtextxy(36*x,11*y,"Mary");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(60*x,9*y,0,359,2);      /* Zamphir */
moveto(55*x,7*y);  lineto(60*x,9*y);
outtextxy(56*x,19*y/2,"Zamphir");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(15*x,9*y,0,359,2);      /* Chris   */
moveto(25*x,7*y);  lineto(15*x,9*y);
outtextxy(9*x,9*y,"Chris");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
pieslice(20*x,11*y,0,359,2);     /* Denise  */
moveto(20*x,11*y);  lineto(15*x,9*y);
outtextxy(15*x,23*y/2,"Denise");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*******************************************************************/
```

```
pieslice(50*x,9*y,0,359,2);      /* Tony   */
moveto(55*x,7*y); lineto(50*x,9*y);
outtextxy(51*x,9*y,"Tony");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(45*x,11*y,0,359,2);     /* Quincy */
moveto(45*x,11*y); lineto(50*x,9*y);
outtextxy(46*x,11*y,"Quincy");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(50*x,13*y,0,359,2);     /* Tom    */
moveto(45*x,11*y); lineto(50*x,13*y);
outtextxy(48*x,27*y/2,"Tom");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(10*x,11*y,0,359,2);     /* Bill   */
moveto(10*x,11*y); lineto(15*x,9*y);
outtextxy(8*x,23*y/2,"Bill");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(25*x,11*y,0,359,2);     /* Jean   */
moveto(25*x,11*y); lineto(30*x,9*y);
outtextxy(24*x,23*y/2,"Jean");
Pause(30*x,24*y);
```

```
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
pieslice(40*x,13*y,0,359,2);     /* Mona    */
moveto(40*x,13*y);  lineto(35*x,11*y);
outtextxy(37*x,27*y/2,"Mona");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(7*x,14*y,"We now will apply inorder traversal to obtain the sorted
                         list:");
outtextxy(30*x,31*y/2,"Inorder  Listing (Sorted listing)");
moveto(2*x,16*y);  lineto(89*x,16*y);
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x.49*y/2);
setcolor(forecolor);
/*****************************************************************/
outtextxy(3*x,17*y,"Bill,");     /* Visit Bill */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(10*x,11*y);  lineto(15*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(10*x,11*y);  lineto(15*x,9*y);
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(9*x,17*y,"Chris,");     /* Visit Chris */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(20*x,11*y);  lineto(15*x,9*y);
```

```
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(20*x,11*y); lineto(15*x,9*y);
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(16*x,17*y,"Denise,");    /* Visit Denise */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(25*x,7*y); lineto(15*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,7*y); lineto(15*x,9*y);
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(24*x,17*y,"Jack,");    /* Visit Jack */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(25*x,7*y); lineto(30*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,7*y); lineto(30*x,9*y);
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(30*x,17*y,"Jean,");    /* Visit Jean */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(25*x,11*y); lineto(30*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,11*y); lineto(30*x,9*y);
setlinestyle(0,0,3);
/*********************************************************************/
outtextxy(36*x,17*y,"John,");    /* Visit John */
```

```
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(35*x,11*y); lineto(30*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,11*y); lineto(30*x,9*y);
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(42*x,17*y,"Mary,");     /* Visit Mary */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(35*x,11*y); lineto(40*x,13*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(35*x,11*y); lineto(40*x,13*y);
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(48*x,17*y,"Mona,");     /* Visit Mona */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(25*x,7*y); lineto(40*x,5*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(25*x,7*y); lineto(40*x,5*y);
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(54*x,17*y,"Quincy");     /* Visit Quincy */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(45*x,11*y); lineto(50*x,13*y);
setlinestyle(3,0,3);
setcolor(forecolor);
```

```
moveto(45*x,11*y); lineto(50*x,13*y);
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(62*x,17*y,"Tom");     /* Visit Tom */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(45*x,11*y); lineto(50*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(45*x,11*y); lineto(50*x,9*y);
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(67*x,17*y,"Tony");     /* Visit Tony */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(55*x,7*y); lineto(50*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,7*y); lineto(50*x,9*y);
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(73*x,17*y,"Vanna");     /* Visit Vanna */
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,50*x,49*y/2);
moveto(55*x,7*y); lineto(60*x,9*y);
setlinestyle(3,0,3);
setcolor(forecolor);
moveto(55*x,7*y); lineto(60*x,9*y);
setlinestyle(0,0,3);
/*******************************************************************/
outtextxy(81*x,17*y,"Zamphir");     /* Visit Zamphir */
Pause(30*x,24*y);
setcolor(backcolor);
```

```
   bar(29*x,23*y,50*x,49*y/2);
   moveto(55*x,7*y);  lineto(40*x,5*y);
   setlinestyle(3,0,3);
   setcolor(forecolor);
   moveto(55*x,7*y);  lineto(40*x,5*y);
   setlinestyle(0,0,3);
   /*******************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field again */
   bar(2*x,17*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}

/*********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,22*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,22*y,179*x/2,49*y/2);
      setcolor(forecolor);
    }
    switch (ch)          {
     case 'y': in_the_exercise = 0;
             brea!:;
     case 'Y': in_the_exercise = 0;
             break;
    case 'n': setcolor(backcolor);
          bar(46*x,35*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    case 'N': setcolor(backcolor);
          bar(46*x,35*y/2,179*x/2,22*y);
          setcolor(forecolor);
          break;

    default : break;
    }
}
```

```
/* PROGRAM   : lang.c
   AUTHOR    : Atilla BAKAN
   DATE      : Mar. 17, 1990
   REVISED   : Apr. 7, 1990

   DESCRIPTION : This program contains the tutorial for an application
                 of trees, namely language syntax.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                          /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                           /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif

#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions           */
static void add_shadow    (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit    (int errnum);
static void initialize    (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void Pageup        (void);
static void Pagedown      (void);
static void press_a_key   (int wrow);
static void pre_help      (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);

/* Tutorial procedures                 */
static void language    (void);
static void ex_lang_1   (void);
static void grammar     (void);
static void backus      (void);
static void ex_lang_2   (void);
static void exercises   (void);
static void exer1       (void);
static void exer2       (void);
static void exer3       (void);
static void final_cut   (void);
static void P1          (void);
static void P2      (void);
static void P3      (void);
static void P4      (void);
static void P5      (void);
static void P6      (void);
static void P7      (void);
static void P8      (void);
static void P9      (void);
static void P10      (void);

/**********************************************************************/
/* miscellaneous global variables                                     */
/**********************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];

/**********************************************************************/
/* error message table                                                */
/**********************************************************************/
static char *error_text[]= {
  NULL,  /* errnum =  0, no error   */
  NULL,  /* errnum == 1, windowing error */
  "Syntax:  CXLDEMO [-switches]\n\n"
     "\t -c = CGA snow elimination\n"
     "\t -b = BIOS screen writing\n"
     "\t -m = force monochrome text attributes",
  "Memory allocation error"
};

/**********************************************************************/
/* miscellaneous defines                                              */
/**********************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33
/**********************************************************************/
```

```
/* this function will add a shadow to the active window                      */
/****************************************************************************/
static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}

/****************************************************************************/
/* this function pops open a window and confirms that the user really        */
/* wants to quit the demo.  If so, it terminates the demo program.           */
/****************************************************************************/
static void confirm_quit(void)
{
   struct _onkey_t *kblist;

   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
   add_shadow();
   wputs("\n Quit demo, are you sure? \033A\156Y\b");
   clearkeys();
   showcur();
   if(wgetchf("YN",'Y')=='Y') normal_exit();
   wclose();
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);       /* restore any hidden hot keys */
}

/****************************************************************************/
/* this function is called by the pull-down demo for a prompt                */
/****************************************************************************/
static void disp_sure_msg(void)
{
   wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
```

```
/****************************************************************************/
/* this function handles abnormal termination.  If it is passed an         */
/* error code of 1, then it is a windowing system error.  Otherwise        */
/* the error message is looked up in the error message table.              */
/****************************************************************************/
static void error_exit(int errnum)
{
   if(errnum) {
      printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
               exit(errnum);
   }
}



/****************************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems */
/****************************************************************************/
static void initialize(void)
{
   /* initialize the CXL video system and save current screen info */
   videoinit();
   readcur(&crow,&ccol);
   if((savescrn=ssave())==NULL) error_exit(3);

   /* if mouse exists, turn on full mouse support */
   if(msinit()) {
      mssupport(MS_FULL);
      msgotoxy(12,49);
   }

   /* attach [Alt-X] to the confirm_quit() function */
   setonkey(0x2d00,confirm_quit,0);

   /* attach [Ctrl Pageup] to the Pageup() function */
   setonkey(0x8400,Pageup,0);
```

```
  /* attach [Ctrl Pagedown] to the Pagedown() function */
  setonkey(0x7600,Pagedown,0);

  /* initialize help system, help key = [F1] */
  whelpdef("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
           WHITE|_RED,RED|_LGREY,pre_help);
}

/**********************************************************************/
static void pre_help(void)
{
  add_shadow();
  setonkey(0x2d00,confirm_quit,0);
}

/**********************************************************************/
/* this function is called anytime to switch back to previous window.       */
/**********************************************************************/
static void Pageup(void)
{
  static WINDOW handle;

  handle = whandle();
  wactiv(handle - 1);
}

/**********************************************************************/
/* this function is called anytime to switch back to next window.           */
/**********************************************************************/
static void Pagedown(void)
{
  static WINDOW handle;

  handle = whandle();
  wactiv(handle + 1);
}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
/*************************************************************************/
/* this function displays a pause message then pauses for a keypress          */
/*************************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;

   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"           ");
   wprints(wrow,0,attr2,"         ");
}
/*************************************************************************/
/* This routine causes short dealys during execution                          */
/*************************************************************************/
static void short_delay(void)
{
   delay_(SHORT_DELAY);
}
```

```
/*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.             */
/*************************************************************************/
static void quit_window(void)
{
   static WINDOW handle=0;

   if(handle) {
      wactiv(handle);
      wclose();
      handle=0;
   }
   else {
      handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
      wputs(" Quit takes you back to the\n demo program's main menu.");
   }
}

/*************************************************************************/
/* shows the cursor again if it has been hidden                          */
/*************************************************************************/
static void restore_cursor(void)
{
   wtextattr(WHITE|_MAGENTA);
   showcur();
}

/*************************************************************************/
/* enlarges or shrinks the windows                                       */
/*************************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
```

```
/*********************************************************************/
/* moves the active window to a given screen coordinates            */
/*********************************************************************/
static void move_window(int nsrow,int nscol)
{
  if(wmove(nsrow,nscol)) error_exit(1);
  short_delay();
}

/*********************************************************************/
/* this routine calls language() routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                  */
/*********************************************************************/
void P1()
{
  wcloseall();
  language();
}
/*********************************************************************/
/* this routine calls ex_lang_1() routine whenever Pageup or         */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P2()
{
  wcloseall();
  ex_lang_1();
}
/*********************************************************************/
/* this routine calls grammar() routine whenever Pageup or           */
/* Pagedown keys are pressed.                                        */
/*********************************************************************/
void P3()
{
  wcloseall();
  grammar();
}
```

```
/**********************************************************************/
/* this routine   calls backus() routine whenever Pageup or           */
/* Pagedown keys are pressed.                                          */
/**********************************************************************/
void P4()
{
   wcloseall();
   backus();
}
/**********************************************************************/
/* this routine  calls ex_lang_2 routine whenever Pageup or           */
/* Pagedown keys are pressed.                                          */
/**********************************************************************/
void P5()
{
   wcloseall();
   ex_lang_2();
}
/**********************************************************************/
/* this routine  calls sorting routine whenever Pageup or             */
/* Pagedown keys are pressed.                                          */
/**********************************************************************/
void P6()
{
   wcloseall();
   exercises();
}
/**********************************************************************/
/* this routine  calls exer1 routine whenever Pageup or               */
/* Pagedown keys are pressed.                                          */
/**********************************************************************/
void P7()
{
   wcloseall();
   exer1();
}
```

```
/*********************************************************************/
/* this routine calls exer2 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                       */
/*********************************************************************/
void P8()
{
   wcloseall();
   exer2();
}
/*********************************************************************/
/* this routine calls exer3 routine whenever Pageup or              */
/* Pagedown keys are pressed.                                       */
/*********************************************************************/
void P9()
{
   wcloseall();
   exer3();
}
/*********************************************************************/
/* this routine calls final_cut() routine whenever Pageup or        */
/* Pagedown keys are pressed.                                       */
/*********************************************************************/
void P10()
{
   wcloseall();
   final_cut();
}

/*********************************************************************/
/* main routine, calls minimal spanning tree tutorial                */
/*********************************************************************/
void main()
{
  initialize();
  language();
}
```

```
/***********************************************************************/
/*This routine calls definition, example and algorithm routines about     */
/* language syntax.                                                        */
/***********************************************************************/
static void language(void)
{
  register int *scrn;

  if((scrn=ssave())==NULL) error_exit(3);
  cclrscrn(LGREY|_BLUE);
  /*******************************************************************/
  /* attach [Pagedown] to the ex_lang_1() function */
  setonkey(0x5100,P2,0);
  /*******************************************************************/
  if((w[1]=wopen(5,15,11,54,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Syntax of Languages]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" English grammar provides a set of rules by which we can"
         " classify words in a sentence according to their function"
         " in the sentence.");
  press_a_key(4);
  wslide(0,0);
  /*******************************************************************/
  if((w[2]=wopen(5,15,16,54,3,LCYAN|_BLACK,BLACK|_GREEN))==0)
          error_exit(1);
  wtitle("[Syntax of Languages]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" In the sentence, The dog chases the cat, 'The dog' is the subject"
         " and 'chases the cat' is the predicate. The subject, in turn,"
         " consists of the article 'the'  followed by the noun 'dog'."
         " The predicate can also be decomposed into the verb 'chases'"
         " and the object 'the cat', which is an article and a noun.");
  press_a_key(9);
  wslide(0,40);
  short_delay();
  ex_lang_1();
  srestore(scrn);
}

/**********************************************************************/
/* This routine gives an example for a parse tree                    */
/**********************************************************************/
static void ex_lang_1 (void)
{
  /********************************************************************/
  /* attach [Pageup] to the language() function */
  setonkey(0x4900,P1,0);
  /********************************************************************/
  /* attach [Pagedown] to the grammar() function */
  setonkey(0x5100,P3,0);
  /********************************************************************/
  if((w[3]=wopen(11,15,16,65,3,LCYAN|_BLACK,BLACK|_MAGENTA))==0)
          error_exit(1);
  wtitle("[Syntax of Languages]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" We can represent this structure by a tree known as a"
          " parse tree. You will see an example in the following "
         " figure.");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"examp471.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  grammar();
}
```

```
/***********************************************************************/
/* This routine talks about the grammar of languages.                 */
/***********************************************************************/
static void grammar(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the ex_lang_1() function */
  setonkey(0x4900,P2,0);
  /*********************************************************************/
  /* attach [Pagedown] to the backus() function */
  setonkey(0x5100,P4,0);
  /*********************************************************************/
  if((w[1]=wopen(5,20,13,60,3,WHITE|_BLACK,RED|_CYAN))==0) error_exit(1);
  wtitle("[Grammars]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" The grammar of a language consists of a set of rules that"
          " specify precisely what descendent nodes each nonterminal"
          " may have. Since the rules can be used to produce sentences,"
          " they are known as production rules.");
  press_a_key(6);
  wslide(0,0);
  short_delay();
  /*********************************************************************/
  if((w[2]=wopen(5,20,18,60,3,WHITE|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Grammars]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" The grammar corresponding to the parse tree that we have"
      " shown to you  is given below :");
  wputs("\n    Sentence  ::=  Subject Predicate");
  wputs("\n    Subject   ::=  Article Noun");
  wputs("\n    Predicate ::=  Verb Object");
  wputs("\n    Object    ::=  Article Noun");
  wputs("\n    Article   ::=  the");
wputs("\n    Noun    ::= dog");
wputs("\n    Noun    ::= cat");
wputs("\n    Verb    ::= chases");
press_a_key(11);
wslide(9,0);
short_delay();
/*********************************************************************/
if((w[3]=wopen(5,20,13,59,3,WHITE|_BLACK,RED|_CYAN))==0) error_exit(1);
wtitle("[Grammars]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw(" The ::= symbol indicates that, on a parse tree, the item on"
          " the right as descendents. One symbol on the left of a ::="
          " will appear as the root of the parse tree. This symbol is"
          " refered to as the start symbol.");
press_a_key(6);
wslide(0,40);
short_delay();
/*********************************************************************/
if((w[4]=wopen(5,20,21,60,3,WHITE|_BLACK,BLACK|_CYAN))==0)
           error_exit(1);
wtitle("[Grammars]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw("  The rules of the grammar can be used to generate all possible"
           " sentences of the language. In our example,since the only"
           " possible terminal symbols are 'the','dog','cat',and"
           " 'chases', there are only four possible parse trees. They"
           " correspond to the following sentences :");
wputs("\n    The dog chases the cat");
wputs("\n    The dog chases the dog");
wputs("\n    The cat chases the dog");
wputs("\n    The cat chases the cat\n");
wputsw("  These sentences are the entire language defined by this"
           " grammar.");
press_a_key(14);
```

```
  wslide(9,39);
  short_delay();
  /*****************************************************************/
  if((w[5]=wopen(5,15,15,65,3,WHITE|_BLACK,WHITE|_BLUE))==0)
            error_exit(1);
  wtitle("[Grammars]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("   Another way to use the rules is to analyze a sentence to see"
            " if it is a syntactically correct sentence in the language."
            " This is done by using the rules to attempt to generate a"
            " parse tree whose terminal symbols are the sentence. If such"
            " a tree can be constructed, the sentence is  part of the"
            " language. The analysis of the sentence in this way is called"
            " parsing.");
  press_a_key(8);
  short_delay();
  wcloseall();
  backus();
}

/*****************************************************************/
/* This routine talks about Backus-Naur Form.                     */
/*****************************************************************/
static void backus(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the grammar() function */
  setonkey(0x4900,P3,0);
  /*****************************************************************/
  /* attach [Pagedown] to the ex_lang_2() function */
  setonkey(0x5100,P5,0);
  /*****************************************************************/
  if((w[1]=wopen(5,20,15,60,3,WHITE|_BLACK,RED|_CYAN))==0) error_exit(1);
  wtitle("[Backus-Naur Form]",TCENTER,_LGREY|BROWN);
  add_shadow();
```

```
whelpcat(H_WINTITLE);
wputsw("  Computer scientists use grammars to give formal definitions"
          " to programming languages. The formal definitions specify the"
          " 'legal' statements that are part of a program written in a"
          " given language. But computer scientists need a language to"
         " define programming languages.");
press_a_key(8);
wslide(0,0);
short_delay();
/*********************************************************************/
if((w[2]=wopen(5,20,13,60,3,WHITE|_BLACK,WHITE|_MAGENTA))==0)
           error_exit(1);
wtitle("[Backus-Naur Form]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw("  A language used to describe other languages is called a"
          " 'metalanguage'. Probably the most common metalanguage used"
          " by computer scientists to define programming languages is"
          " known as 'Backus-Naur Form', abbriviated BNF.");
press_a_key(6);
wslide(0,40);
short_delay();
/*********************************************************************/
if((w[3]=wopen(5,10,14,65,3,WHITE|_BLACK,BLACK|_CYAN))==0)
           error_exit(1);
wtitle("[Grammars]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputsw(" For example, the BNF specification of a grammar for simple"
           " arithmatic expression may appear as :");
wputs("\n <Expression> ::= <Expression> <Operator> <Expression>");
wputs("              | (<Expression>) ");
wputs("\n              | - <Expression>) ");
wputs("\n              |0|1|2|3|4|5|6|7|8|9 ");
wputs("\n <Operator>   ::= +|-|*|/|^ ");
press_a_key(7);
```

```
   wslide(9,0);
   short_delay();
   /************************************************************************/
   if((w[4]=wopen(5,10,19,65,3,WHITE|_CYAN,RED|_BLACK))==0) error_exit(1);
   wtitle("[Grammars]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputsw(" In this grammar, the nonterminal symbols are <Expression>"
            " and <Operator>. <Expression> is the start symbol. The terminal"
           " symbols are ");
   wputs("\n  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *, /, ^, (, and ).\n");
   wputsw("  Note that the definition of this language is recursive"
           " because <Expression> is defined in terms of itself.");
   wputs("\n  The symbols of BNF are :");
   wputs("\n    ::=  'is defined as'");
   wputs("\n     |    'or'");
   wputs("\n    < >  'syntactic category name'");
   press_a_key(12);
   short_delay();
   wcloseall();
   ex_lang_2();
}


/************************************************************************/
/* This routine gives an example for Backus-Naur Form                   */
/************************************************************************/
static void ex_lang_2 (void)
{
   /************************************************************************/
   /* attach [Pageup] to the backus() function */
   setonkey(0x4900,P4,0);
   /************************************************************************/
   /* attach [Pagedown] to the final_cut() function */
   setonkey(0x5100,P10,0);
   /************************************************************************/
   if((w[2]=wopen(5,15,10,65,3,WHITE|_BLACK,WHITE|_LGREY))==0)
            error_exit(1);
   wtitle("[Binary Search Trees - Example_4_1]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n");
   wputsw("  Let's illustrate this grammar with an example ?");
   press_a_key(3);
   short_delay();
   wclose();
   spawnl(P_WAIT,"examp472.exe",NULL);
   cclrscrn(LGREY|_BLUE);
   final_cut();
}

/*************************************************************************/
/* This routine finishes the session with language syntax               */
/*************************************************************************/
static void final_cut(void)
{
   /*************************************************************************/
   /* attach [Pageup] to the ex_lang_2() function */
   setonkey(0x4900,P5,0);
   /*************************************************************************/
   /* attach [Pagedown] to the exercises() function */
   setonkey(0x5100,P6,0);
   /*************************************************************************/
   if((w[1]=wopen(5,20,12,60,3,WHITE|_BLACK,WHITE|_MAGENTA))==0)
            error_exit(1);
   wtitle("[Backus-Naur Form]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputsw("  There is another method of defining a programming language."
          " which is called 'syntax diagrams'. But this method is not"
          " in our scop that's why we will not cover this topic.");
   press_a_key(5);
```

```
    short_delay();
    wclose();
    exercises();
}

/************************************************************************/
/* This routine makes a small quiz about the language syntax.          */
/************************************************************************/
void exercises(void)
 {
    register int *screen;

    /*********************************************************************/
    /* attach [Pageup] to the final_cut() function */
    setonkey(0x4900,P10,0);
    /*********************************************************************/
    /* attach [Pagedown] to the exer1() function */
    setonkey(0x5100,P7,0);
    /*********************************************************************/
    if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
               error_exit(1);
    wtitle("[Language Syntax]",TCENTER,_LGREY|BROWN);
    whelpcat(H_WINTITLE);
    add_shadow();
    wputs("\n");
    wputsw(" We have completed our presentation of this section. Are"
              " you ready for a pop quiz ? ");
    press_a_key(3);
    short_delay();
    wclose();
    if((screen=ssave())==NULL) error_exit(3); {
    /*********************************************************************/
     exer1();
    /*********************************************************************/
    /* if mouse exists, turn on full mouse support */
     if(msinit()) {
        mssupport(MS_FULL);
        msgotoxy(12,49);
        }
    }
    srestore(screen);
}







/*********************************************************************/
/* Dummy function to call the actual exercise 4.7.1                  */
/*********************************************************************/
static void exer1(void)
{
    /*****************************************************************/
    /* attach [Pageup] to the final_cut() function      */
    setonkey(0x4900,P10,0);
    /*****************************************************************/
    /* attach [Pagedown] to the exer2() function */
    setonkey(0x5100,P8,0);
    /*****************************************************************/
    if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
            error_exit(1);
    wtitle("[Language Syntax]",TCENTER,_LGREY|BROWN);
    whelpcat(H_WINTITLE);
    add_shadow();
    wputs("\n");
    wputsw("      Here is the first question. ");
    press_a_key(3);
    wclose();
    spawnl(P_WAIT,"q471.exe",NULL);
    cclrscrn(LGREY|_BLUE);
    exer2();
}
```

```
/***********************************************************************/
/* Dummy function to call the actual exercise 4.7.2                    */
/***********************************************************************/
static void exer2(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the exer1() function          */
  setonkey(0x4900,P7,0);
  /*********************************************************************/
  /* attach [Pagedown] to the exer3() function */
  setonkey(0x5100,P9,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
        error_exit(1);
  wtitle("[Language Syntax]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the second question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q472.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer3();
}
```

```
/***********************************************************************/
/* Dummy function to call the actual exercise 4.7.3                    */
/*********************************************** *********************/
static void exer3(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the exer2() function             */
  setonkey(0x4900,P8,0);
  /*********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
        error_exit(1);
  wtitle("[Language Syntax]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("     Here is the third question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q473.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  normal_exit();
}
```

```
/* PROGRAM    : examp471.c
   AUTHOR      : Atilla BAKAN
   DATE        : Apr. 18, 1990
   REVISED     : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for a parse tree.



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"

#if defined(__TURBOC__)                          /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                       /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                         /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif

#define _GRAPH_T_DEFINED
```

```
/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);



/*********************************************************************/
/* graphic initialization variables                                  */
/*********************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;



/*********************************************************************/
/* This function is used for including drivers to the executable code  */
/*********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/***********************************************************************/
/* This fuction initializes the necessary graphical routines          */
/***********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
       }
  forecolor = WHITE;
 }
```

```
/*********************************************************************/
/* This function sets the text default values                         */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*********************************************************************/
/* Equivalent of press_a_key function for graphics screen            */
/*********************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
  }



/*********************************************************************/
/* main routine, calls exer routine                                   */
/*********************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates a parse tree.                            */
/*********************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-7-1");
  /*****************************************************************/
  pieslice(45*x,2*y,0,359,2);  /* Sentence    */
  pieslice(25*x,5*y,0,359,2);  /* Subject     */
  pieslice(65*x,5*y,0,359,2);  /* Predicate   */
  moveto(25*x,5*y);  lineto(45*x,2*y);  lineto(65*x,5*y);
  outtextxy(40*x,3*y/2,"sentence");
  outtextxy(16*x,5*y,"subject");
  outtextxy(67*x,5*y,"predicate");
  /*****************************************************************/
  pieslice(15*x,10*y,0,359,2);
  pieslice(35*x,10*y,0,359,2);
  pieslice(50*x,10*y,0,359,2);
  pieslice(70*x,10*y,0,359,2);
  moveto(15*x,10*y);  lineto(25*x,5*y); lineto(35*x,10*y);
  moveto(50*x,10*y);  lineto(65*x,5*y); lineto(70*x,10*y);
  outtextxy(6*x,10*y,"article");
  outtextxy(37*x,10*y,"noun");
  outtextxy(52*x,10*y,"verb");
  outtextxy(72*x,10*y,"object");
  /*****************************************************************/
  pieslice(60*x,13*y,0,359,2);
  pieslice(80*x,13*y,0,359,2);
  moveto(60*x,13*y);  lineto(70*x,10*y); lineto(80*x,13*y);
  outtextxy(62*x,13*y,"article");
  outtextxy(82*x,13*y,"noun");
  /*****************************************************************/
  pieslice(15*x,16*y,0,359,2);
  pieslice(35*x,16*y,0,359,2);
  pieslice(50*x,16*y,0,359,2);
  pieslice(60*x,16*y,0,359,2);
  pieslice(80*x,16*y,0,359,2);
  moveto(15*x,10*y);  lineto(15*x,16*y);
  moveto(35*x,10*y);  lineto(35*x,16*y);
  moveto(50*x,10*y);  lineto(50*x,16*y);
  moveto(60*x,13*y);  lineto(60*x,16*y);
  moveto(80*x,13*y);  lineto(80*x,16*y);
  outtextxy(13*x,33*y/2,"The");
  outtextxy(33*x,33*y/2,"dog");
  outtextxy(47*x,33*y/2,"chases");
  outtextxy(58*x,33*y/2,"the");
  outtextxy(78*x,33*y/2,"cat");
  /*****************************************************************/
   outtextxy(2*x,18*y,"There are two kinds of nodes in the parse tree. One kind rep-
resents the");
    outtextxy(2*x,19*y,"words of the original English sentence. These nodes appear
as the leaves of");
    outtextxy(2*x,20*y,"the tree and are called terminals. The terminals, taken in left
to right or-");
   outtextxy(2*x,21*y,"der, form the original English sentence. The other kind repre-
sents grammati-");
    outtextxy(2*x,22*y,"cal categories. These nodes are called nonterminals, or syn-
tactic categories.");
  /*****************************************************************/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
}
```

```
/* PROGRAM   : examp472.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 18, 1990
   REVISED    : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for backus naur form.



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"

#if defined(__TURBOC__)                       /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)           /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                       /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif

#define _GRAPH_T_DEFINED
```

```
/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;




/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/***********************************************************************/
/* This fuction initializes the necessary graphical routines           */
/***********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
     }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
     }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
    }
  }



/*************************************************************************/
/* main routine, calls exer routine                                      */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*********************************************************************/
/* This routine illustrates an example of Backus-Naur form.          */
/*********************************************************************/
void exer()
{

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-7-2");
  /*****************************************************************/
  pieslice(45*x,2*y,0,359,2);   /* Expression   */
  pieslice(25*x,5*y,0,359,2);   /* Expression   */
  pieslice(45*x,5*y,0,359,2);   /* Operator     */
  pieslice(65*x,5*y,0,359,2);   /* Expression   */
  moveto(25*x,5*y); lineto(45*x,2*y); lineto(65*x,5*y);
  moveto(45*x,5*y); lineto(45*x,2*y);
  outtextxy(39*x,3*y/2,"expression");
  outtextxy(12*x,5*y,"expression");
  outtextxy(47*x,5*y,"operator");
  outtextxy(67*x,5*y,"expression");
  /*****************************************************************/
  pieslice(15*x,10*y,0,359,2);
  pieslice(25*x,10*y,0,359,2);
  pieslice(35*x,10*y,0,359,2);
  pieslice(45*x,10*y,0,359,2);
  pieslice(65*x,10*y,0,359,2);
  moveto(15*x,10*y); lineto(25*x,5*y); lineto(35*x,10*y);
  moveto(25*x,10*y); lineto(25*x,5*y);
  moveto(45*x,10*y); lineto(45*x,5*y);
  moveto(65*x,10*y); lineto(65*x,5*y);
  outtextxy(15*x,21*y/2,"(");
  outtextxy(20*x,21*y/2,"expression");
  outtextxy(35*x,21*y/2,")");
  outtextxy(45*x,21*y/2,"-");
```

```
    outtextxy(65*x,21*y/2,"1");
    /*********************************************************************/
    pieslice(17*x,13*y,0,359,2);
    pieslice(25*x,13*y,0,359,2);
    pieslice(33*x,13*y,0,359,2);
    moveto(17*x,13*y); lineto(24*x,11*y);
    moveto(25*x,13*y); lineto(25*x,11*y);
    moveto(33*x,13*y); lineto(26*x,11*y);
    outtextxy(8*x,27*y/2,"expression");
    outtextxy(21*x,27*y/2,"operator");
    outtextxy(31*x,27*y/2,"expression");
    /*********************************************************************/
    pieslice(17*x,16*y,0,359,2);
    pieslice(25*x,16*y,0,359,2);
    pieslice(33*x,16*y,0,359,2);
    moveto(17*x,14*y); lineto(17*x,16*y);
    moveto(25*x,14*y); lineto(25*x,16*y);
    moveto(33*x,14*y); lineto(33*x,16*y);
    outtextxy(17*x,33*y/2,"5");
    outtextxy(25*x,33*y/2,"^");
    outtextxy(33*x,33*y/2,"2");
    /*********************************************************************/
    outtextxy(8*x,18*y,"The parse tree above shows that the expression");
    outtextxy(8*x,19*y,"          ( 5 ^ 2 ) - 1");
    outtextxy(8*x,20*y,"is a valid expressiom in this language. But the expression");
    outtextxy(8*x,21*y,"            5 + ^ 2 ");
    outtextxy(8*x,22*y,"is not valid in this language. ( Why ? )");
    /*********************************************************************/
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM     : q471.c
   AUTHOR      : Atilla BAKAN
   DATE        : Apr. 6, 1990
   REVISED     : Apr. 6, 1990

   DESCRIPTION : This program contains the first exercise about the binary
                 trees and traversals.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a.c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk         find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name         name
   #define ff_attrib      attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions        */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);
static void reason1         (void);
static void reason2         (void);
static void reason3         (void);

/*****************************************************************/
/* miscellaneous global variables                               */
/*****************************************************************/
 int in_the_exercise = 1;




/*****************************************************************/
/* graphic initialization variables                             */
/*****************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/************************************************************************/
/* This function is used for including drivers to the executable code      */
/************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/************************************************************************/
/* This fuction initializes the necessary graphical routines              */
/************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /**********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /**********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /**********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /**********************************************************************/
  settext();
  /**********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
    else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
   forecolor = WHITE;
  }

/****************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL); /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,59*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)         {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
     case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
     case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}




/***********************************************************************/
/* This function sets the text default values                          */
/***********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*******************************************************************/
/* Equivalent of press_a_key function for graphics screen          */
/*******************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }

/*******************************************************************/
/* main routine, calls exer routine                                */
/*******************************************************************/
void main()
{
  exer();
}

/*******************************************************************/
/* This routine that asks the question, then depending on the user's answer */
/* makes necessary explanations                                    */
/*******************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  1");
   outtextxy(18*x,2*y,"Consider the following grammar for expressions.");
   /***************************************************************/
   outtextxy(20*x,4*y,"<Expression> ::= <Term> | <Expression> + <Term>");
   outtextxy(20*x,5*y,"<Term>       ::= <Operand> | <Term> * <Operand>");
outtextxy(20*x,6*y,"<Operand>   ::=  A | B | C");
/*******************************************************************/
outtextxy(18*x,12*y,"Which one of the following expressions is illegal");
outtextxy(18*x,13*y,"according to the grammar above ?");
outtextxy(20*x,15*y,"a)  A");
outtextxy(20*x,16*y,"b)  A * B * C + A ");
outtextxy(20*x,17*y,"c)  C + B * A");
outtextxy(20*x,18*y,"d)  (A * B) + C");
outtextxy(18*x,20*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,20*y,"    Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(50*x,19*y,88*x,21*y);
   setcolor(forecolor);
}
 switch (Ch)          {
  case 'a': outtextxy(50*x,20*y,"a");
       outtextxy(55*x,20*y,"No, You are wrong! Because");
       outtextxy(55*x,21*y,"A is a legal expression. Look");
       outtextxy(55*x,22*y,"at the following parse tree.");
       outtextxy(55*x,23*y,"(The correct answer is 'd')");
       Pause(30*x,24*y);
       reason1();
       break;
  case 'b': outtextxy(50*x,20*y,"b");
       outtextxy(55*x,20*y,"Sorry, You are wrong! Look");
       outtextxy(55*x,21*y,"at the following parse tree");
       outtextxy(55*x,22*y,"to see why.");
       outtextxy(55*x,23*y,"(The correct answer is 'd')");
       Pause(30*x,24*y);
       reason2();
```

```
            break;
        case 'c': outtextxy(50*x,20*y,"c");
            outtextxy(55*x,20*y,"Sorry, You are wrong! Look");
            outtextxy(55*x,21*y,"at the following parse tree");
            outtextxy(55*x,22*y,"to see why.");
            outtextxy(55*x,23*y,"(The correct answer is 'd')");
            Pause(30*x,24*y);
            reason3();
            break;
        case 'd': outtextxy(50*x,20*y,"d");
             outtextxy(55*x,21*y,"Yes. You are right.");
             outtextxy(55*x,22*y,"Congratulations!");
             Pause(30*x,24*y);
             break;
        default  : break;
        }
        closegraph();
}

/*****************************************************************/
/* This routine gives reasoning to the first choice             */
/*****************************************************************/
static void reason1(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,13*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
   /**************************************************************/
   pieslice(45*x,7*y,0,359,2);   /* Expression    */
   pieslice(45*x,9*y,0,359,2);   /* Term          */
   pieslice(45*x,11*y,0,359,2);  /* Operand       */
   pieslice(45*x,13*y,0,359,2);  /*    A          */
   moveto(45*x,7*y);  lineto(45*x,9*y);
   lineto(45*x,11*y); lineto(45*x,13*y);
   outtextxy(46*x,7*y,"Expression");
   outtextxy(46*x,9*y,"Term");
   outtextxy(46*x,11*y,"Operand");
   outtextxy(45*x,27*y/2,"A");
   /*******************************************************************/
     outtextxy(4*x,15*y,"As you see  A  is a legal expression with respect to the this
grammar.");
   /*******************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);         /* Clean the game field  again */
   bar(2*x,13*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
}

/*******************************************************************/
/* This routine gives reasoning to the second choice                */
/*******************************************************************/
static void reason2(void)
{
   setcolor(backcolor);         /* Clean the game field */
   bar(2*x,13*y/2,179*x/2,49*y/2);
   setcolor(forecolor);
   /*******************************************************************/
   pieslice(45*x,7*y,0,359,2);  /* Expression   */
   pieslice(35*x,8*y,0,359,2);  /* Expression   */
   pieslice(45*x,8*y,0,359,2);  /*    +       */
   pieslice(55*x,8*y,0,359,2);  /* Term        */
   moveto(35*x,8*y); lineto(45*x,7*y);
   lineto(55*x,8*y);
   moveto(45*x,7*y); lineto(45*x,8*y);
   outtextxy(46*x,7*y,"Expression");
   outtextxy(23*x,8*y,"Expression");
   outtextxy(45*x,17*y/2,"+");
   outtextxy(56*x,8*y,"Term");
   /*******************************************************************/
   pieslice(35*x,9*y,0,359,2);  /* Term   */
   pieslice(55*x,9*y,0,359,2);  /* Operand */
   moveto(35*x,8*y); lineto(35*x,9*y);
moveto(55*x,8*y); lineto(55*x,9*y);
outtextxy(29*x,9*y,"Term");
outtextxy(56*x,9*y,"Operand");
/*****************************************************************/
pieslice(25*x,10*y,0,359,2);  /* Term    */
pieslice(35*x,10*y,0,359,2);  /*    *    */
pieslice(45*x,10*y,0,359,2);  /* Operand */
pieslice(55*x,10*y,0,359,2);  /*    A    */
moveto(25*x,10*y); lineto(35*x,9*y);
moveto(35*x,10*y); lineto(35*x,9*y);
moveto(45*x,10*y); lineto(35*x,9*y);
moveto(55*x,10*y); lineto(55*x,9*y);
outtextxy(19*x,10*y,"Term");
outtextxy(35*x,21*y/2,"*");
outtextxy(44*x,19*y/2,"Operand");
outtextxy(55*x,21*y/2,"A");
/*****************************************************************/
pieslice(20*x,11*y,0,359,2);  /* Term    */
pieslice(25*x,11*y,0,359,2);  /*    *    */
pieslice(30*x,11*y,0,359,2);  /* Operand */
pieslice(45*x,11*y,0,359,2);  /*    C    */
moveto(25*x,10*y); lineto(20*x,11*y);
moveto(25*x,10*y); lineto(25*x,11*y);
moveto(25*x,10*y); lineto(30*x,11*y);
moveto(45*x,10*y); lineto(45*x,11*y);
outtextxy(14*x,11*y,"Term");
outtextxy(25*x,23*y/2,"*");
outtextxy(31*x,11*y,"Operand");
outtextxy(45*x,23*y/2,"C");
/*****************************************************************/
pieslice(20*x,12*y,0,359,2);  /* Operand */
pieslice(30*x,12*y,0,359,2);  /*    B    */
moveto(20*x,12*y); lineto(20*x,11*y);
moveto(30*x,12*y); lineto(30*x,11*y);
outtextxy(11*x,12*y,"Operand");
outtextxy(30*x,25*y/2,"B");
```

```
  /***********************************************************************/
  pieslice(20*x,13*y,0,359,2);   /* A */
  moveto(20*x,12*y);  lineto(20*x,13*y);
  outtextxy(20*x,27*y/2,"A");
  /***********************************************************************/
  outtextxv(4*x,15*y,"As you see  we can represent each of the elements in the
                        expression");
  outtextxy(4*x,16*y,"as a terminals. So  A * B * C + A is a legal expression.");
  /***********************************************************************/
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field  again */
  bar(2*x,13*y/2,179*x/2,49*y/2);
  setcolor(forecolor);
}

/***********************************************************************/
/* This routine gives reasoning to the third choice                    */
/***********************************************************************/
static void reason3(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,13*y/2,179*x/2,49*y/2);
  setcolor(forecolor);
  /***********************************************************************/
  pieslice(45*x,7*y,0,359,2);   /* Expression   */
  pieslice(35*x,8*y,0,359,2);   /* Expression   */
  pieslice(45*x,8*y,0,359,2);   /*    +        */
  pieslice(55*x,8*y,0,359,2);   /* Term         */
  moveto(35*x,8*y);  lineto(45*x,7*y);
  lineto(55*x,8*y);
  moveto(45*x,7*y);  lineto(45*x,8*y);
  outtextxy(46*x,7*y,"Expression");
  outtextxy(23*x,8*y,"Expression");
  outtextxy(45*x,17*y/2,"+");
  outtextxy(56*x,8*y,"Term");
  pieslice(35*x,9*y,0,359,2);   /* Term   */
```

```
    pieslice(50*x,9*y,0,359,2);   /* Term    */
    pieslice(55*x,9*y,0,359,2);   /*  *     */
    pieslice(60*x,9*y,0,359,2);   /* Operand */
    moveto(35*x,8*y);  lineto(35*x,9*y);
    moveto(55*x,8*y);  lineto(50*x,9*y);
    moveto(55*x,8*y);  lineto(55*x,9*y);
    moveto(55*x,8*y);  lineto(60*x,9*y);
    outtextxy(29*x,9*y,"Term");
    outtextxy(44*x,9*y,"Term");
    outtextxy(55*x,19*y/2,"*");
    outtextxy(61*x,9*y,"Operand");
    /*****************************************************************/
    pieslice(35*x,10*y,0,359,2);   /* Operand   */
    pieslice(50*x,10*y,0,359,2);   /* Operand   */
    pieslice(60*x,10*y,0,359,2);   /*  A        */
    moveto(35*x,10*y);  lineto(35*x,9*y);
    moveto(50*x,10*y);  lineto(50*x,9*y);
    moveto(60*x,10*y);  lineto(60*x,9*y);
    outtextxy(26*x,10*y,"Operand");
    outtextxy(41*x,10*y,"Operand");
    outtextxy(60*x,21*y/2,"A");
    /*****************************************************************/
    pieslice(35*x,11*y,0,359,2);   /* C */
    pieslice(50*x,11*y,0,359,2);   /* B */
    moveto(35*x,10*y);  lineto(35*x,11*y);
    moveto(50*x,10*y);  lineto(50*x,11*y);
    outtextxy(35*x,23*y/2,"C");
    outtextxy(50*x,23*y/2,"B");
    outtextxy(4*x,15*y,"As you see  we can represent each of the elements in the
                              expression");
    outtextxy(4*x,16*y,"as a terminals. So  C + B * A  is a legal expression.");
    Pause(30*x,24*y);
    setcolor(backcolor);          /* Clean the game field  again */
    bar(2*x,13*y/2,179*x/2,49*y/2);
    setcolor(forecolor);
}
```

```
/* PROGRAM  : q472.c
   AUTHOR    : Atilla BAKAN
   DATE       : Apr. 6, 1990
   REVISED   : Apr. 6, 1990

   DESCRIPTION : This program contains the second exercise about the
                 language syntax.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                 /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions          */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer             (void);
static void step_solution     (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;



/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*****************************************************************************/
/* This function is used for including drivers to the executable code       */
/*****************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*****************************************************************************/
/* This fuction initializes the necessary graphical routines                 */
/*****************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***************************************************************************/
  settext();
  /***************************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/**********************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)         {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) mscshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/*****************************************************************************/
/* This function sets the text default values                               */
/*****************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/***********************************************************************/
/* Equivalent of press_a_key function for graphics screen              */
/***********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }

/***********************************************************************/
/* main routine, calls exer routine                                    */
/***********************************************************************/
void main()
{
  exer();
}

/***********************************************************************/
/* This routine  asks the question, then depending on the user's answer */
/* makes necessary explanations                                         */
/***********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  2");
   outtextxy(18*x,2*y,"Consider the following grammar for expressions.");
   /*******************************************************************/
   outtextxy(20*x,3*y,"<Expression> ::= <Term> | <Expression> + <Term>");
   outtextxy(20*x,4*y,"<Term>       ::= <Operand> | <Term> * <Operand>");
outtextxy(20*x,5*y,"<Operand>   ::=  A | B | C");
/*****************************************************************/
outtextxy(14*x,7*y,"Construct the parse tree for the following  expression.");
outtextxy(35*x,8*y,"A * B + C * A");
/*****************************************************************/
while (in_the_exercise == 1) {
outtextxy(15*x,14*y,"Choose one of the following, as you need :");
outtextxy(15*x,15*y,"    a) I'm done, I want to compare my solution with yours.");
outtextxy(15*x,16*y,"    b) I want to see step by step solution.");
outtextxy(15*x,17*y,"    c) This is enough for me, I want to exit.");
outtextxy(15*x,18*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c'))) {
    outtextxy(48*x,18*y,"    Please type a, b, or c");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c')) {
    setcolor(backcolor);
    bar(50*x,35*y/2,88*x,20*y);
    setcolor(forecolor);
    }
  }
    switch (Ch)         {
   case 'a': outtextxy(47*x,18*y,"a");
     outtextxy(52*x,18*y,"You want to compare your solu-");
     outtextxy(52*x,19*y,"tion with ours. So press any  ");
     outtextxy(52*x,20*y,"key to see it.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,35*y/2,179*x/2,22*y);
     setcolor(forecolor);
     compare_solutions();
     break;
   case 'b': outtextxy(47*x,18*y,"b");
     outtextxy(52*x,18*y,"You want to see step by step");
```

```
        outtextxy(52*x,19*y,"solution. So press any key to ");
        outtextxy(52*x,20*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,35*y/2,179*x/2,22*y);
        setcolor(forecolor);
        step_solution();
        break;
      case 'c': outtextxy(47*x,18*y,"c");
        confirm_exit();
        break;
      default  : break;
     }
   }
   closegraph();
 }

/*******************************************************************/
/* This routine gives the solution to the exercise to be compared.         */
/*******************************************************************/
static void compare_solutions(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,33*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   pieslice(45*x,9*y,0,359,2);   /* Expression    */
   pieslice(35*x,10*y,0,359,2);  /* Expression    */
   pieslice(45*x,10*y,0,359,2);  /*    +        */
   pieslice(55*x,10*y,0,359,2);  /* Term         */
   moveto(35*x,10*y);  lineto(45*x,9*y);
   lineto(55*x,10*y);
   moveto(45*x,9*y);  lineto(45*x,10*y);
   outtextxy(46*x,9*y,"Expression");
   outtextxy(23*x,10*y,"Expression");
   outtextxy(45*x,21*y/2,"+");
outtextxy(56*x,10*y,"Term");
/*******************************************************************/
pieslice(35*x,11*y,0,359,2);  /* Term    */
pieslice(55*x,11*y,0,359,2);  /* Term    */
pieslice(50*x,11*y,0,359,2);  /*  *      */
pieslice(60*x,11*y,0,359,2);  /* Operand */
moveto(35*x,10*y); lineto(35*x,11*y);
moveto(55*x,10*y); lineto(50*x,11*y);
moveto(55*x,10*y); lineto(55*x,11*y);
moveto(55*x,10*y); lineto(60*x,11*y);
outtextxy(29*x,11*y,"Term");
outtextxy(44*x,11*y,"Term");
outtextxy(55*x,23*y/2,"*");
outtextxy(61*x,11*y,"Operand");
/*******************************************************************/
pieslice(30*x,12*y,0,359,2);  /* Term    */
pieslice(35*x,12*y,0,359,2);  /*  *      */
pieslice(40*x,12*y,0,359,2);  /* Operand */
pieslice(50*x,12*y,0,359,2);  /* Operand */
pieslice(60*x,12*y,0,359,2);  /*  A      */
moveto(30*x,12*y); lineto(35*x,11*y);
moveto(35*x,12*y); lineto(35*x,11*y);
moveto(40*x,12*y); lineto(35*x,11*y);
moveto(50*x,12*y); lineto(50*x,11*y);
moveto(60*x,12*y); lineto(60*x,11*y);
outtextxy(24*x,12*y,"Term");
outtextxy(35*x,25*y/2,"*");
outtextxy(39*x,23*y/2,"Operand");
outtextxy(51*x,12*y,"Operand");
outtextxy(60*x,25*y/2,"A");
/*******************************************************************/
pieslice(30*x,13*y,0,359,2);  /* Operand */
pieslice(40*x,13*y,0,359,2);  /*  B      */
pieslice(50*x,13*y,0,359,2);  /*  C      */
moveto(30*x,12*y); lineto(30*x,13*y);
moveto(40*x,12*y); lineto(40*x,13*y);
```

```
   moveto(50*x,12*y); lineto(50*x,13*y);
   outtextxy(21*x,13*y,"Operand");
   outtextxy(40*x,27*y/2,"B");
   outtextxy(50*x,27*y/2,"C");
   /*****************************************************************/
   pieslice(30*x,14*y,0,359,2);   /* A */
   moveto(30*x,14*y); lineto(30*x,13*y);
   outtextxy(30*x,29*y/2,"A");
   /*****************************************************************/
   Pause(30*x,24*y);
   setcolor(backcolor);          /* Clean the game field  again */
   bar(2*x,33*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
}

/*******************************************************************/
/* This routine gives the step by step solution to the exercise              */
/*******************************************************************/
static void step_solution(void)
{
   setcolor(backcolor);          /* Clean the game field */
   bar(2*x,33*y/4,179*x/2,49*y/2);
   setcolor(forecolor);
   /*****************************************************************/
   pieslice(45*x,9*y,0,359,2);   /* Expression    */
   outtextxy(46*x,9*y,"Expression");
   Pause(30*x,24*y);
   /*****************************************************************/
   pieslice(35*x,10*y,0,359,2);   /* Expression    */
   pieslice(45*x,10*y,0,359,2);   /*    +          */
   pieslice(55*x,10*y,0,359,2);   /* Term          */
   moveto(35*x,10*y); lineto(45*x,9*y);
   lineto(55*x,10*y);
   moveto(45*x,9*y); lineto(45*x,10*y);
   outtextxy(23*x,10*y,"Expression");
   outtextxy(45*x,21*y/2,"+");
outtextxy(56*x,10*y,"Term");
Pause(30*x,24*y);
/*******************************************************************/
pieslice(35*x,11*y,0,359,2);  /* Term    */
pieslice(55*x,11*y,0,359,2);  /* Term    */
pieslice(50*x,11*y,0,359,2);  /*  *      */
pieslice(60*x,11*y,0,359,2);  /* Operand */
moveto(35*x,10*y); lineto(35*x,11*y);
moveto(55*x,10*y); lineto(50*x,11*y);
moveto(55*x,10*y); lineto(55*x,11*y);
moveto(55*x,10*y); lineto(60*x,11*y);
outtextxy(29*x,11*y,"Term");
outtextxy(44*x,11*y,"Term");
outtextxy(55*x,23*y/2,"*");
outtextxy(61*x,11*y,"Operand");
Pause(30*x,24*y);
/*******************************************************************/
pieslice(30*x,12*y,0,359,2);  /* Term    */
pieslice(35*x,12*y,0,359,2);  /*  *      */
pieslice(40*x,12*y,0,359,2);  /* Operand */
pieslice(50*x,12*y,0,359,2);  /* Operand */
pieslice(60*x,12*y,0,359,2);  /*  A      */
moveto(30*x,12*y); lineto(35*x,11*y);
moveto(35*x,12*y); lineto(35*x,11*y);
moveto(40*x,12*y); lineto(35*x,11*y);
moveto(50*x,12*y); lineto(50*x,11*y);
moveto(60*x,12*y); lineto(60*x,11*y);
outtextxy(24*x,12*y,"Term");
outtextxy(35*x,25*y/2,"*");
outtextxy(39*x,23*y/2,"Operand");
outtextxy(51*x,12*y,"Operand");
outtextxy(60*x,25*y/2,"A");
Pause(30*x,24*y);
/*******************************************************************/
pieslice(30*x,13*y,0,359,2);  /* Operand */
pieslice(40*x,13*y,0,359,2);  /*  B      */
```

```
  pieslice(50*x,13*y,0,359,2);   /*  C    */
  moveto(30*x,12*y);  lineto(30*x,13*y);
  moveto(40*x,12*y);  lineto(40*x,13*y);
  moveto(50*x,12*y);  lineto(50*x,13*y);
  outtextxy(21*x,13*y,"Operand");
  outtextxy(40*x,27*y/2,"B");
  outtextxy(50*x,27*y/2,"C");
  Pause(30*x,24*y);
  /*****************************************************************/
  pieslice(30*x,14*y,0,359,2);   /* A */
  moveto(30*x,14*y);  lineto(30*x,13*y);
  outtextxy(30*x,29*y/2,"A");
  /*****************************************************************/
  Pause(30*x,24*y);
  setcolor(backcolor);           /* Clean the game field  again */
  bar(2*x,33*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}

/*******************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,22*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,21*y,179*x/2,49*y/2);
      setcolor(forecolor);
    }
    switch (ch)       {
     case 'y': in_the_exercise = 0;
            break;
     case 'Y': in_the_exercise = 0;
            break;

     case 'n': setcolor(backcolor);
            bar(46*x,35*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;

     case 'N': setcolor(backcolor);
            bar(46*x,35*y/2,179*x/2,22*y);
            setcolor(forecolor);
            break;

     default : break;
     }
}
```

```
/* PROGRAM   : q473.c
   AUTHOR     : Atilla BAKAN
   DATE       : Apr. 7, 1990
   REVISED    : Apr. 7, 1990

   DESCRIPTION : This program contains the third exercise about the
                 language syntax.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)       _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)      _dos_findnext(a)
   #define ffblk            find_t
   #define ff_name          name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk            FIND
   #define ff_name          name
   #define ff_attrib        attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions     */
static void exer            (void);
static void example          (void);
static void show_alg         (void);
static void step_solution    (void);
static void compare_solutions (void);
static void confirm_exit      (void);

/************************************************************************/
/* miscellaneous global variables                                        */
/************************************************************************/
 int in_the_exercise = 1;


/************************************************************************/
/* graphic initialization variables                                      */
/************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/**********************************************************************/
/* This function is used for including drivers to the executable code        */
/**********************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/**********************************************************************/
/* This fuction initializes the necessary graphical routines                  */
/**********************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /********************************************************************/
  settext();
  /********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
   ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
    setfillstyle(SOLID_FILL,BLACK);
    backcolor = BLACK;
    quitcolor = WHITE;
    }
  else {
    setfillstyle(SOLID_FILL,BLUE);
    backcolor = BLUE;
    quitcolor = RED;
    }
  forecolor = WHITE;
 }

/**********************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
```

```
          videoinit();
          exit(0);
          break;
    case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
    case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    case 'N': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);    /* restore any hidden hot keys */
}



/*********************************************************************/
/* This function sets the text default values                        */
/*********************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/**********************************************************************/
/* Equivalent of press_a_key function for graphics screen           */
/**********************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }

/**********************************************************************/
/* main routine, calls exer routine                                   */
/**********************************************************************/
void main()
{
  exer();
}

/**********************************************************************/
/* This routine  asks the question, then depending on the user's answer */
/* makes necessary explanations                                        */
/**********************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  3");
   outtextxy(18*x,2*y,"Consider the following grammar for expressions.");
   /******************************************************************/
   outtextxy(20*x,3*y,"<Expression> ::= <Term> | <Expression> + <Term>");
   outtextxy(20*x,4*y,"<Term>       ::= <Operand> | <Term> * <Operand>");
outtextxy(20*x,5*y,"<Operand>    ::=  A | B | C");
/**********************************************************************/
outtextxy(14*x,7*y,"Construct the parse tree for the following  expression.");
outtextxy(35*x,8*y,"B + B + C + A");
/**********************************************************************/
while (in_the_exercise == 1) {
outtextxy(14*x,14*y,"Choose one of the following, as you need :");
outtextxy(14*x,15*y,"     a) I'm done, I want to compare my solution with yours.");
outtextxy(14*x,16*y,"     b) I want to see step by step solution.");
outtextxy(14*x,17*y,"     c) This is enough for me, I want to exit.");
outtextxy(15*x,18*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c'))) {
    outtextxy(48*x,18*y,"    Please type a, b, or c");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'a') || (Ch == 'b') || (Ch == 'c')) {
    setcolor(backcolor);
    bar(50*x,35*y/2,88*x,20*y);
    setcolor(forecolor);
    }
   }
    switch (Ch)          {
   case 'a': outtextxy(47*x,18*y,"a");
     outtextxy(52*x,18*y,"You want to compare your solu-");
     outtextxy(52*x,19*y,"tion with ours. So press any  ");
     outtextxy(52*x,20*y,"key to see it.");
     Pause(30*x,24*y);
     setcolor(backcolor);
     bar(50*x,35*y/2,179*x/2,22*y);
     setcolor(forecolor);
     compare_solutions();
     break;
   case 'b': outtextxy(47*x,18*y,"b");
     outtextxy(52*x,18*y,"You want to see step by step")
```

```
        outtextxy(52*x,19*y,"solution. So press any key to ");
        outtextxy(52*x,20*y,"continue.");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(50*x,35*y/2,179*x/2,22*y);
        setcolor(forecolor);
        step_solution();
        break;
      case 'c': outtextxy(47*x,18*y,"c");
        confirm_exit();
        break;
      default  : break;
    }
  }
  closegraph();
}

/*************************************************************************/
/* This routine gives the solution to the exercise to be compared.       */
/*************************************************************************/
static void compare_solutions(void)
{

  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,33*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  /**********************************************************************/
  pieslice(45*x,9*y,0,359,2);    /* Expression   */
  pieslice(35*x,10*y,0,359,2);   /* Expression   */
  pieslice(45*x,19*y/2,0,359,2); /*    +         */
  pieslice(55*x,10*y,0,359,2);   /* Term         */
  moveto(35*x,10*y);  lineto(45*x,9*y);
  lineto(55*x,10*y);
  moveto(45*x,9*y);  lineto(45*x,19*y/2);
  outtextxy(46*x,9*y,"Expression");
  outtextxy(22*x,10*y,"Expression");
outtextxy(45*x,10*y,"+");
outtextxy(56*x,10*y,"Term");
/*******************************************************************/
pieslice(25*x,11*y,0,359,2);  /* Expression   */
pieslice(35*x,21*y/2,0,359,2); /*    +        */
pieslice(45*x,11*y,0,359,2);  /* Term         */
pieslice(55*x,11*y,0,359,2);  /* Operand      */
moveto(25*x,11*y); lineto(35*x,10*y);
lineto(45*x,11*y);
moveto(35*x,10*y); lineto(35*x,21*y/2);
moveto(55*x,10*y); lineto(55*x,11*y);
outtextxy(12*x,11*y,"Expression");
outtextxy(56*x,11*y,"Operand");
outtextxy(35*x,11*y,"+");
outtextxy(46*x,11*y,"Term");
/*******************************************************************/
pieslice(20*x,12*y,0,359,2);  /* Expression   */
pieslice(25*x,23*y/2,0,359,2); /*    +        */
pieslice(30*x,12*y,0,359,2);  /* Term         */
pieslice(45*x,12*y,0,359,2);  /* Operand      */
pieslice(55*x,12*y,0,359,2);  /*    A         */
moveto(20*x,12*y); lineto(25*x,11*y);
lineto(30*x,12*y);
moveto(25*x,11*y); lineto(25*x,23*y/2);
moveto(45*x,11*y); lineto(45*x,12*y);
moveto(55*x,11*y); lineto(55*x,12*y);
outtextxy(7*x,12*y,"Expression");
outtextxy(46*x,12*y,"Operand");
outtextxy(25*x,12*y,"+");
outtextxy(31*x,12*y,"Term");
outtextxy(55*x,25*y/2,"A");
/*******************************************************************/
pieslice(20*x,13*y,0,359,2);  /* Term   */
pieslice(30*x,13*y,0,359,2);  /* Operand */
pieslice(45*x,13*y,0,359,2);  /*   C   */
moveto(20*x,12*y); lineto(20*x,13*y);
```

```
  moveto(30*x,12*y); lineto(30*x,13*y);
  moveto(45*x,12*y); lineto(45*x,13*y);
  outtextxy(31*x,13*y,"Operand");
  outtextxy(14*x,13*y,"Term");
  outtextxy(45*x,27*y/2,"C");
  /*****************************************************************/
  pieslice(20*x,14*y,0,359,2);  /* Operand */
  pieslice(30*x,14*y,0,359,2);  /*   B    */
  moveto(20*x,13*y); lineto(20*x,14*y);
  moveto(30*x,13*y); lineto(30*x,14*y);
  outtextxy(11*x,14*y,"Operand");
  outtextxy(30*x,29*y/2,"B");
  /*****************************************************************/
  pieslice(20*x,15*y,0,359,2);  /* B */
  moveto(20*x,15*y); lineto(20*x,14*y);
  outtextxy(20*x,31*y/2,"B");
  /*****************************************************************/
  Pause(30*x,24*y);
  setcolor(backcolor);          /* Clean the game field  again */
  bar(2*x,33*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
}

/*******************************************************************/
/* This routine gives the step by step solution to the exercise          */
/*******************************************************************/
static void step_solution(void)
{
  setcolor(backcolor);          /* Clean the game field */
  bar(2*x,33*y/4,179*x/2,49*y/2);
  setcolor(forecolor);
  /*****************************************************************/
  pieslice(45*x,9*y,0,359,2);  /* Expression    */
  outtextxy(46*x,9*y,"Expression");
  Pause(30*x,24*y);
```

```
pieslice(35*x,10*y,0,359,2);   /* Expression    */
pieslice(45*x,19*y/2,0,359,2); /*    +          */
pieslice(55*x,10*y,0,359,2);   /* Term          */
moveto(35*x,10*y); lineto(45*x,9*y);
lineto(55*x,10*y);
moveto(45*x,9*y); lineto(45*x,19*y/2);
outtextxy(46*x,9*y,"Expression");
outtextxy(22*x,10*y,"Expression");
outtextxy(45*x,10*y,"+");
outtextxy(56*x,10*y,"Term");
Pause(30*x,24*y);
/*****************************************************************/
pieslice(25*x,11*y,0,359,2);   /* Expression    */
pieslice(35*x,21*y/2,0,359,2); /*    +          */
pieslice(45*x,11*y,0,359,2);   /* Term          */
pieslice(55*x,11*y,0,359,2);   /* Operand       */
moveto(25*x,11*y); lineto(35*x,10*y);
lineto(45*x,11*y);
moveto(35*x,10*y); lineto(35*x,21*y/2);
moveto(55*x,10*y); lineto(55*x,11*y);
outtextxy(12*x,11*y,"Expression");
outtextxy(56*x,11*y,"Operand");
outtextxy(35*x,11*y,"+");
outtextxy(46*x,11*y,"Term");
Pause(30*x,24*y);
/*****************************************************************/
pieslice(20*x,12*y,0,359,2);   /* Expression    */
pieslice(25*x,23*y/2,0,359,2); /*    +          */
pieslice(30*x,12*y,0,359,2);   /* Term          */
pieslice(45*x,12*y,0,359,2);   /* Operand       */
pieslice(55*x,12*y,0,359,2);   /*    A          */
moveto(20*x,12*y); lineto(25*x,11*y);
lineto(30*x,12*y);
moveto(25*x,11*y); lineto(25*x,23*y/2);
moveto(45*x,11*y); lineto(45*x,12*y);
moveto(55*x,11*y); lineto(55*x,12*y);
```

```
    outtextxy(7*x,12*y,"Expression");
    outtextxy(46*x,12*y,"Operand");
    outtextxy(25*x,12*y,"+");
    outtextxy(31*x,12*y,"Term");
    outtextxy(55*x,25*y/2,"A");
    Pause(30*x,24*y);
/***********************************************************************/
    pieslice(20*x,13*y,0,359,2);   /* Term    */
    pieslice(30*x,13*y,0,359,2);   /* Operand */
    pieslice(45*x,13*y,0,359,2);   /*   C     */
    moveto(20*x,12*y);  lineto(20*x,13*y);
    moveto(30*x,12*y);  lineto(30*x,13*y);
    moveto(45*x,12*y);  lineto(45*x,13*y);
    outtextxy(31*x,13*y,"Operand");
    outtextxy(14*x,13*y,"Term");
    outtextxy(45*x,27*y/2,"C");
    Pause(30*x,24*y);
/***********************************************************************/
    pieslice(20*x,14*y,0,359,2);   /* Operand */
    pieslice(30*x,14*y,0,359,2);   /*   B     */
    moveto(20*x,13*y);  lineto(20*x,14*y);
    moveto(30*x,13*y);  lineto(30*x,14*y);
    outtextxy(11*x,14*y,"Operand");
    outtextxy(30*x,29*y/2,"B");
    Pause(30*x,24*y);
/***********************************************************************/
    pieslice(20*x,15*y,0,359,2);   /* B */
    moveto(20*x,15*y);  lineto(20*x,14*y);
    outtextxy(20*x,31*y/2,"B");
/***********************************************************************/
    Pause(30*x,24*y);
    setcolor(backcolor);           /* Clean the game field  again */
    bar(2*x,33*y/4,179*x/2,49*y/2);
    setcolor(forecolor);

}
```

```
/***********************************************************************/
static void confirm_exit(void)
{
  char ch;

  outtextxy(52*x,18*y,"You wanted to exit. ");
  outtextxy(52*x,19*y,"Are you sure ? ");
  outtextxy(52*x,20*y,"Type y or n --->");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(53*x,22*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(50*x,21*y,179*x/2,49*y/2);
      setcolor(forecolor);
   }
    switch (ch)        {
    case 'y': in_the_exercise = 0;
           break;
    case 'Y': in_the_exercise = 0;
           break;

    case 'n': setcolor(backcolor);
           bar(46*x,35*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    case 'N': setcolor(backcolor);
           bar(46*x,35*y/2,179*x/2,22*y);
           setcolor(forecolor);
           break;

    default : break;
    }
}
```

LIST OF REFERENCES

1. Tubb, Gary W., *Current Use of Computer in the Teaching of Statistics*, pp. 18 - 20, paper presented at the Computer Science and Statistics Annual Symposium 10th, Gaithersburg, Maryland, 14 April 1977.

2. Hallworth, H. J., and Brebner, Ann, *Computer Assisted Instruction in Schools Achievements, Present Developments and Projections for the Future*, Calgory University (Alberta), sponsored by the Alberta Department of Education, Edmonton, Planning and Research Branch, June 1980.

3. Hirschbuhl, John J. , *Blueprint for the Future of Computer-Based Instruction*, University of Akron, Akron, Ohio, 1977.

4. Harrod, Norma and Ruggles, Marilyn, *Computer Assisted Instruction : An Educational Tool , Focus on Exceptional Children, Vol. 16*, No. 1, September 1983.

5. Office of Naval Research Tech. Memo No. 17, *Current Research Development in Computer Assisted Instruction*, pp. 2, by D. N. Hansen.

BIBLIOGRAPHY

Buake, R. L., *CAI Sourcebook*, Prentice Hall, Inc., Englewood Cliffs, New Jersey 1982.

Dean C. and Whitlock Q. , *A Handbook of Computer Based Training*, Nichols Publishing Company, New York, 1983.

Dossey, J. A., Otto, A. D. , Spence, L. E., and Eynden, C.V., *Discrete Mathematics*, Scott Foresman and Company, Glenview, Illinois, 1987.

Gibbons, A. , *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1984.

Kolman, B. and Busby, R. C. , *Discrete Mathematical Structures for Computer Science*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1984.

Skvarcius, R. and Robinson, W. B. , *Discrete Mathematics with Computer Science Applications*, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1986.

Stevens, R. T. , *Graphics Programming in C*, M&T Publishing, Inc., Redwood City, California, 1988.

Tutte, W. T. , *Graph Theory, Encyclopedia of Mathematics and Its Applications Volume 21*, Addison-Wesley Publishing Company, Menlo Park, California, 1984.

Walther, H. , *Ten Applications of Graph Theory*, D. Reidel Publishing Company, Hingham, MA. 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Department Chairman, Code CS/Mz Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
4.	Prof. Mantak Shing, Code CS/Sh Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
5.	Prof. Kim Hefner, Code MA/Hk Department of Mathematics Naval Postgraduate School Monterey, California 93943	1
6.	Ltjg. Atilla Bakan Topcular mah. Necipokaner sok. 5/5 Rami Istanbul Turkey	4
7.	Ltjg. Yavuz Bas Pazarbasi mah. Simsek sok. 9/17 Uskudar Istanbul Turkey	4

8.	Deniz Harp Okulu Kutuphanesi Deniz Harp Okulu Komutanligi Tuzla Istanbul Turkey	2
9.	Kara Harp Okulu Kutuphanesi Kara Harp Okulu Komutanligi Bakanliklar Ankara Turkey	2
10	Hava Harp Okulu Kutuphanesi Hava Harp Okulu Komutanligi Yesilyurt Istanbul Turkey	2
11	Deniz Lisesi Kutuphanesi Deniz Lisesi Komutanligi Heybeliada Istanbul Turkey	2